A COMPUTER BASED DATA ACQUISITION SYSTEM

FOR THE TEXAS TECH TOKAMAK

by

STEVEN ROBERT BECKERICH, B.S. in E.E.

A THESIS

IN

ELECTRICAL ENGINEERING

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

Approved

Accepted

May 1980

OCU
T3
1980
No.9
cop.2

ACKNOWLEDGEMENTS

I would like to thank Dr. Magne Kristiansen and Dr. Marion Hagler for their support and guidance throughout this endeavor. I would also like to thank Dr. Boyd Blackwell for serving on my committee and for his excellent contributions to this project. I wish to thank Dr. Wayne Ford for serving on my committee and my wife Carol and my sister Sharon for the editing and typing of this manuscript. Finally, I would like to thank my fellow graduate students for their advice and assistance throughout this project.

TABLE OF CONTENTS

ACKNOWLEDGMENTS ii

LIST OF TABLES v

LIST OF FIGURES vi

I. INTRODUCTION 1

II. THE CAMAC STANDARD 6

2.1 Introduction 6

2.2 The CAMAC Crate 7

2.3 The CAMAC Dataway 7

2.4 The Crate Controller 9

2.5 CAMAC Commands 10

2.6 Dataway Timing 10

III. EQUIPMENT 13

3.1 Introduction 13

3.2 The Host Computer 13

3.3 The DL11 Serial Interface 15

3.4 Optical Isolation 17

3.5 The Communications Interface 22

3.6 Terminals 22

3.7 11/04 - CAMAC Interface 25

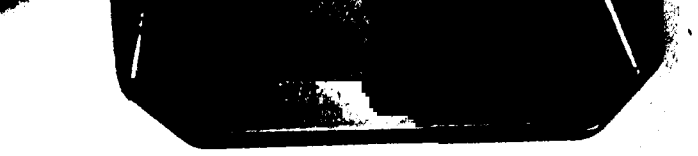
3.8 The DCC-11 Interface 28

3.9 CAMAC Modules 28

IV. SOFTWARE 36

4.1 Introduction 36

4.2 System Information 36



4.3 System Programs 39

4.4 Data Disks 40

4.5 The BASIC Language 41

4.6 OVRNEW.BAS 43

4.7 Hardware Configuration File 45

4.8 PLT200.BAS 47

4.9 STO200.BAS and Support Files 49

4.10 The Data File 57

4.11 Temporary Files 62

4.12 RED200.BAS 63

V. OPERATING INSTRUCTIONS 65

5.1 Introduction 65

5.2 Turn-on and Bootstrap Procedure 65

5.3 RT-11 Commands 66

5.4 Creating a Data Disk 67

5.5 Running the Storage Routines 68

5.6 CAMAC "F" Commands. 69

5.7 System Commands 70

5.8 Recovery From a Crash 79

VI. CONCLUSIONS AND FUTURE IMPROVEMENTS 81

LIST OF REFERENCES 84

APPENDIX A. PROGRAM LISTINGS AND DISCUSSIONS OF STORAGE ROUTINES 85

APPENDIX B. USER FUNCTION ADDITIONS TO BASIC 115

APPENDIX C. STORAGE ROUTINE COMMANDS 122

APPENDIX D. AN EXAMPLE DATA ANALYSIS PROGRAM 126

LIST OF TABLES

Table		Page
1-1	Plasma Parameters for the Texas Tech Tokamak.	3
3-1	Components Included in the 11/04 Computer System	14
4-1	Directory of Storage Routine System Disk and Formatted Data Disk	38
A-1	Entry Points for the Graphics Routines	97
A-2	Subroutines Used in the Storage Routines	108
C-1	Record Titles and Types Contained in TITLE.REC.	124

LIST OF FIGURES

Figure		Page
1-1	Block Diagram of Data Acquisition System	2
2-1	CAMAC Crate with Power Supply and Cooling Fans	8
2-2	Timing Diagram of a Dataway Operation	11
3-1	Remote Halt/Continue Switch and PC Board Modifications .	16
3-2	Front Panel of Terminal Selection Board	18
3-3	Schematic of Terminal Selection Circuit	19
3-4	Continuation of Fig. 3-3	20
3-5	Schematic of Optical Isolation Circuit	21
3-6	Front Panel of Communications Interface	23
3-7	Schematic of Communications Interface	24
3-8	Signals on the Unibus.	26
3-9	Signals on the Unibus	27
3-10	EMI Induced on Unibus During Tokamak Discharge	29
3-11	Schematic of 6 Channel Analog Buffer	31
3-12	Switch Position Readout Logic for 6 Channel Analog Buffer	32
3-13	Block Diagram of 2 Channel Display Driver	34
3-14	Continuation of Fig. 3-13	35
4-1	Flow Diagram of the Initialization Portion of OVRNEW.BAS .	44
4-2(a)	Hardware Configuration File	46
4-2(b)	Typical Format File	46
4-3	Flow Diagram of PLT200.BAS	48
4-4	Flow Diagram of ST0200.BAS	50

Figure Page

4-5 Typical Parameter File 52

4-6 Probe Position and Orientation 55

4-7 Typical Data Header 59

4-8 Flow Diagram of RED200.BAS 64

CHAPTER I

INTRODUCTION

As experiments and diagnostic methods become more sophisticated, the investigator is faced with an ever-increasing amount of data to assimilate and analyze. The data are often in the form of voltage and current waveforms which may not be meaningful until combined with other information or transformed mathematically. Also, data must be archived in such a way as to be easily retrieved for comparison and analysis.

The advent of digital sampling and control equipment combined with the falling prices of small computer systems has greatly reduced the investigator's plight. Technological advances in magnetic media have also made the long term storage of large amounts of digital data both feasible and cost effective. Once stored in digital form, the data can be manipulated quickly by a digital computer. To realize these advantages, a digital data acquisition system, as illustrated in Fig. 1-1, has been developed for the Texas Tech Tokamak.

The Texas Tech Tokamak is a toroidal plasma research device of circular cross section intended primarily for the study of wave-plasma interactions and plasma heating. The facility consists of the toroidal vacuum chamber (R = 46 cm, r = 16 cm), a 130 kJ toroidal field bank, a 20 kJ ohmic heating bank (soon to be increased to 30 kJ), and a 2kJ vertical field bank. Typical plasma parameters are listed in Table 1-1 and extensive documentation of the construction and performance of the Texas Tech Tokamak are available in other reports.[1,2]

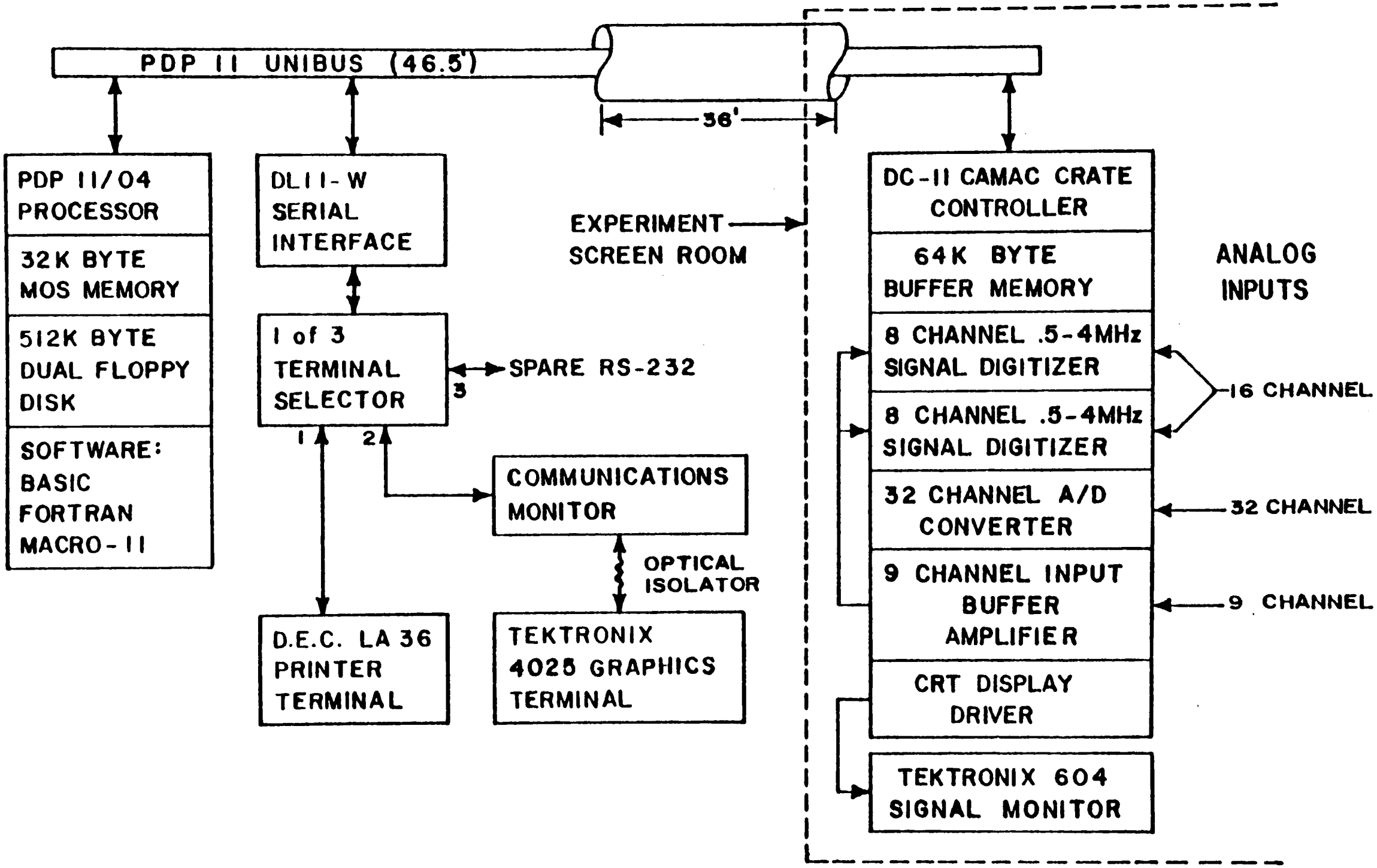


Fig. 1-1. Block Diagram of Data Acquisition System

TABLE 1-1

Plasma Parameters for the Texas Tech Tokamak

Plasma Current	15 kA
Plasma Duration	10 ms
V_{loop}	1.5 V
n_e (max)	1.2×10^{13} cm^{-3}
T_e	100 - 150 eV
T_i	20 - 40 eV
B_ϕ	7 kG

The main criteria when developing the data acquisition system were that it be flexible and easily upgraded as the experimental requirements change. To satisfy these conditions, a Digital Equipment (DEC) PDP-11/04 was selected as the host computer. Due to the widespread use and modular construction of the PDP-11 series computers, a variety of peripheral equipment is available from many vendors. The 11/04 system selected includes 32 k-byte of dynamic RAM and a 512 k-byte dual floppy disk drive. Communication to and from the computer is via a DEC LA36 ASCII printer terminal.

The interface between computer and experiment is accomplished by a CAMAC crate (described in Ch. II) and a Standard Engineering DCC-11 crate controller (described in Ch. III). The name CAMAC is an acronym for Computer Automated Measurement and Control and refers to a set of hardware and software standards which are discussed in detail in Ch. II. In general, a CAMAC system is a self-contained group of digitizing and control equipment which in turn can be controlled from, and transfer data to, a host computer via the crate controller module. In the system described by this report, the CAMAC modules have "local" memory and are capable of storing and displaying the raw data before they are transferred to the host computer.

Data points can be listed in tabular form on the printer or displayed as waveforms on either of two CRT displays. One display is a Tektronix 604 monitor which is interfaced directly to the CAMAC digitizers and is used to review the data presently stored within the digitizers. This monitor allows the user to determine the validity

of the data before using computer time and disk space to store and analyze them. The other CRT display is a Tektronix 4025 graphics terminal which serves both as a graphics display and as a second communications terminal to the computer. With the graphics terminal, data presently stored in the digitizers can be plotted and compared to data previously stored on disks. A hard copy can be made of any graphics or text on the screen of the 4025 terminal via the Tektronix 4631 hard copy unit which connects to the rear of the graphics terminal and can be controlled from either the terminal or the host computer.

In subsequent chapters, the points touched upon in this brief overview will be discussed in more detail. The CAMAC Standard is described in Ch. II and each piece of hardware is discussed in Ch. III. A thorough understanding of the storage system software, while not required to use the system, will be necessary in order to make additions to the routines as requirements change. These programs are discussed in Ch. IV and in Appendix A. Chapter V is intended to be an orderly set of user instructions for the data storage routines to move data to and from disk files. In closing, Ch. VI discusses some logical areas for future expansion and improvement of the system.

CHAPTER II

THE CAMAC STANDARD

Sec. 2.1 Introduction

This chapter is intended as an introduction to the CAMAC Standard in general and not as a description of the data acquisition system discussed in the rest of this report. An understanding of this material is not required to run the present system but may prove helpful to persons intending to specify new CAMAC equipment or those writing control programs for the equipment.

The CAMAC Standard[3] is the result of a combined effort in 1975 by the Nuclear Instrument Module (NIM) Committee of the U.S. Energy Research and Development Administration (now DOE) and the ESONE Committee of European Laboratories.[4] The CAMAC Standards have also been adopted by the Institute of Electrical and Electronics Engineers (IEEE). These standards were developed to simplify the hardware and software efforts involved in interfacing between an experiment or processing system and the diagnostics and control equipment required to monitor the process. Standardization of control and data signals within the CAMAC system, as well as the physical and electrical specifications of the hardware, allows for simple and inexpensive modification and expansion of the system as requirements change.

The individual CAMAC modules can usually work on their own (manual control) but are designed to be computer controlled either from within the CAMAC system or by an external "host" computer. A typical system

would include: a CAMAC crate, a host computer, a crate controller/computer interface, and various CAMAC modules to perform the necessary diagnostic functions.

Sec. 2.2 The CAMAC Crate

The CAMAC crate, as shown in Fig. 2-1, contains, powers, cools, and interconnects up to 25 separate modules. Each module position, or station, is supplied with ± 6 VDC and ± 24 VDC regulated, fused power. These four supplies are monitored by a meter at the front of the crate. The modules are supported by tracks along the top and bottom of the module compartment. Forced air cooling is supplied by fans housed in a separate tray below the module compartment. Additional power supply lines are connected to each station for ± 12 VDC, + 200 VDC, and 117 VAC supplies which are offered as options by many manufacturers but are not required by the CAMAC Standard.

Sec. 2.3 The CAMAC Dataway

All module stations are interconnected at the rear of the module compartment by a bus called the Dataway. The CAMAC Dataway is a multi-layered printed circuit board which supplies power to each station and transfers command, data, status, and address information between stations. Connections to the Dataway are made via an 86 contact PC edge card connector at each of the 25 stations. Stations 1 to 24 are called "normal" stations and will accept any CAMAC module while station 25 is reserved for the crate controller module (discussed in the next section). The normal stations (1-24) share common power supply lines,

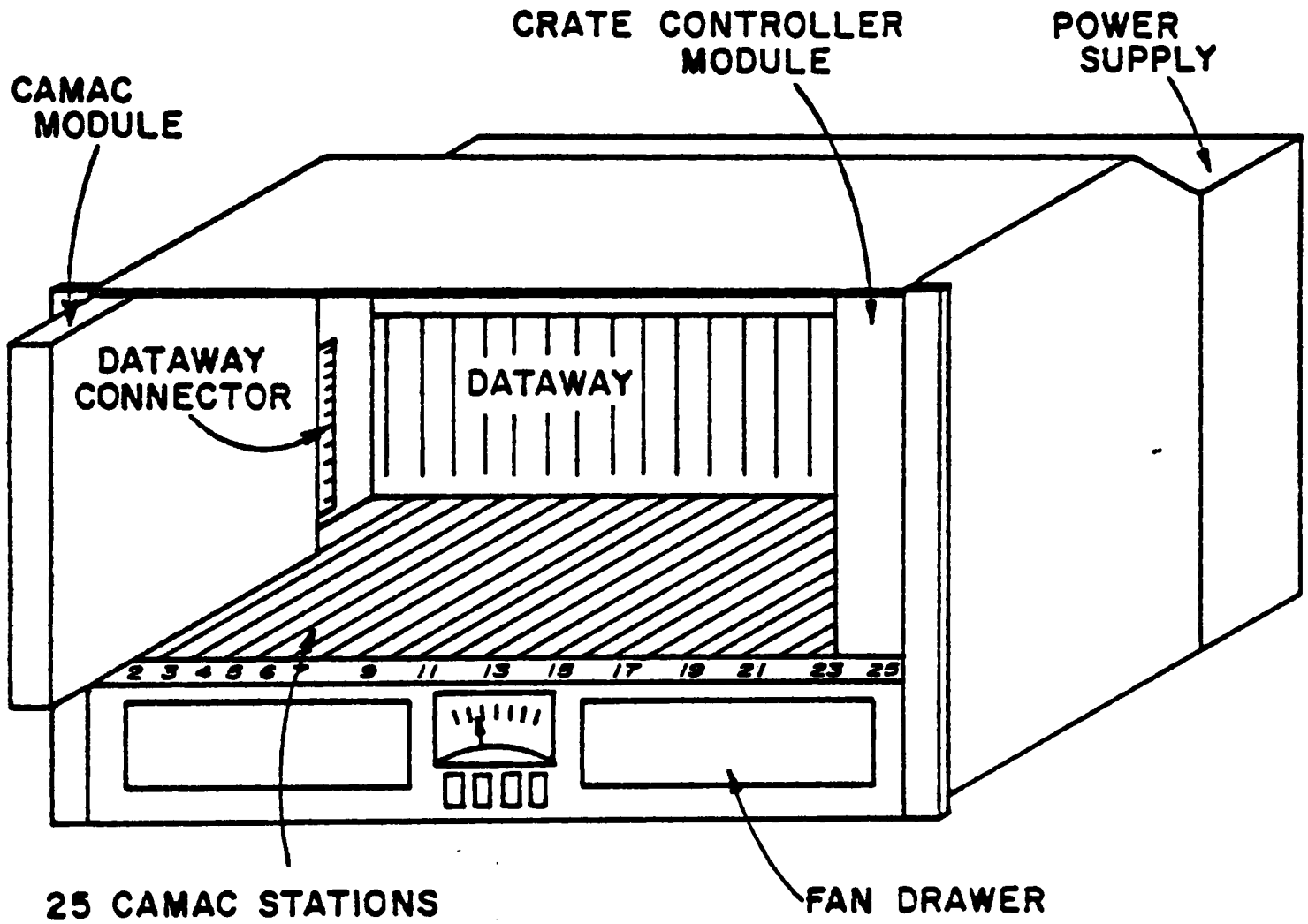


Fig. 2-1. CAMAC Crate with Power Supply and Cooling Fans

subaddress lines, data lines, module status lines, and command lines. Each station has a separate station address line ("N"), and "Look at ME" line ("LAM", often used to request service) that terminates at the controller station (25).

Sec. 2.4 The Crate Controller

The crate controller occupies (at least) the two right-most stations (24 and 25) in the crate. The function of the controller is to initiate and monitor all Dataway activity. A module can request service by raising its "Look at Me" line, and can indicate its status to the controller on the "X" and "Q" status lines. The controller also issues CAMAC commands and acts as a data buffer for transferring data to or from the modules. In selecting a crate controller, there are two major subdivisions to consider; local intelligence vs remote intelligence.

A "smart" crate controller contains local intelligence (usually in the form of a microprocessor) and local memory and is capable of storing and executing programs to control the functions of each module. For communication with devices or computers outside the CAMAC system, a simple parallel or serial (e.g. the "RS-232" standard) interface is usually employed. In contrast to the controller described above, the "dumb" crate controller has little or no local intelligence and is primarily to "translate" commands and data, from an external host computer to the form required by the Dataway and the CAMAC modules.

The type of controller required depends upon the application. Controllers are available which allow several crates to be controlled,

in a serial or parallel arrangement, by a single host computer or smart crate controller. The types of data and their rate of transfer must also be considered when selecting a controller. If a dumb controller is selected, it must be compatible with the bus structure, the input/output conventions, and the signal protocol of the host computer.

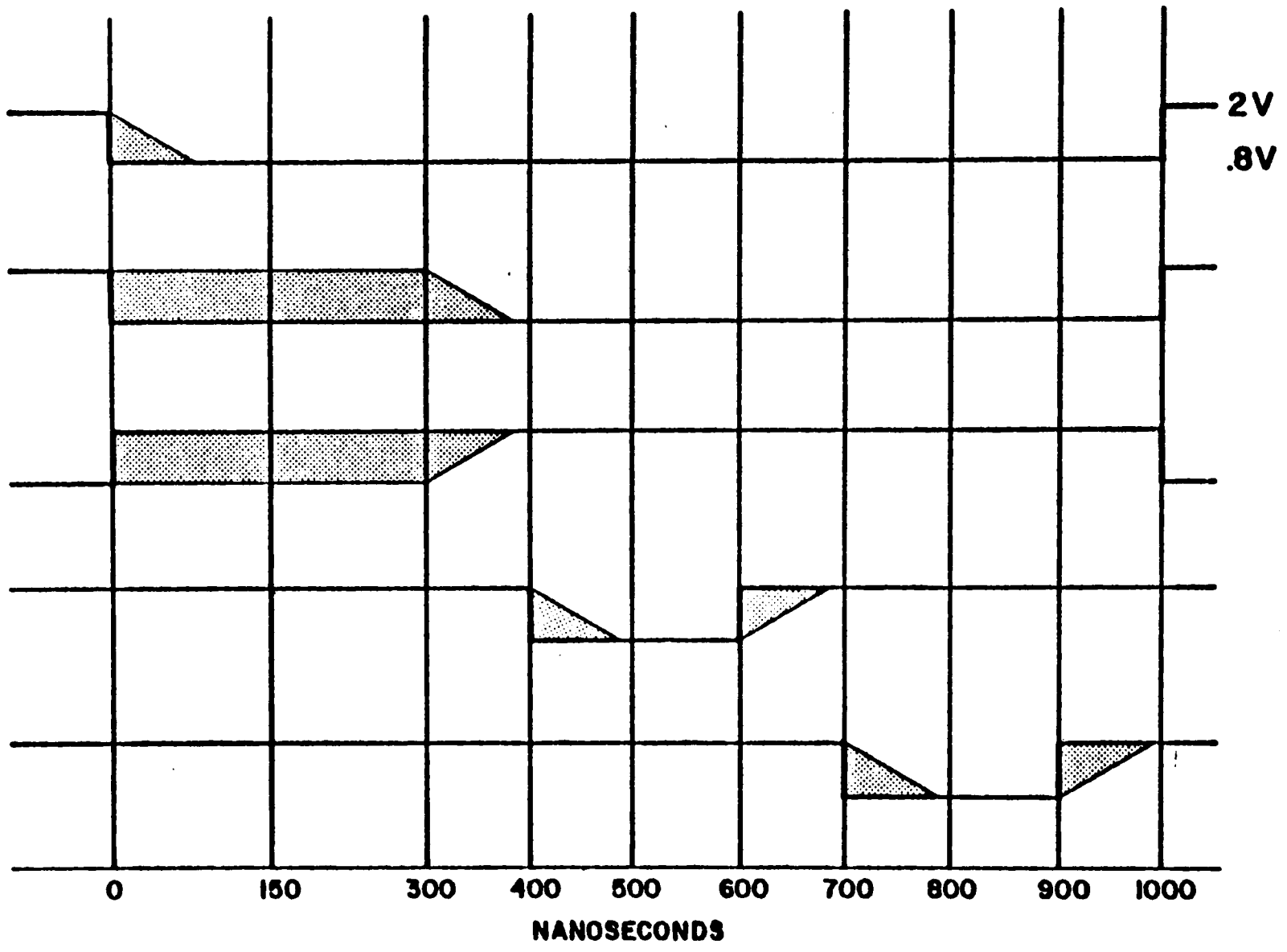
Sec. 2.5 CAMAC Commands

As mentioned in the previous section, the CAMAC modules are controlled via commands issued by the crate controller. Each module is addressed by its station number (N). Within each module, any one of 16 subaddresses can be specified by a 4 bit subaddress code (A_1,A_2,A_4, A_8) generated by the controller. There are 32 CAMAC commands (called "F" codes) which are selected by a 5 bit command code (F_1,F_2,F_4,F_8,F_{16}) and directed to an individual module and subaddress by the (N) and (A) codes. The response from the module to this command depends upon the particular piece of equipment being used (an example of the execution of a CAMAC command is given in the next section).

Sec. 2.6 Dataway Timing

Signals on the Dataway are synchronized by two timing signals (S_1,S_2) which are generated by the crate controller and are common to all stations. A typical Dataway operation is illustrated in Fig. 2-2 and is described in the following example.

Suppose the module in station 13 is a 12 bit analog to digital converter which stores the results in an internal buffer accessed through subaddress A(3). After a conversion the ADC loads its buffer and issues a "Look at Me" signal to the controller. To transfer the



NOTE-

SHADED AREAS INDICATE MAXIMUM ALLOWABLE TOLERANCE FOR RELIABLE OPERATION

Fig. 2-2. Timing Diagram of a Dataway Operation

information from the module, the crate controller would simultaneously clear the "Look at Me" signal, and issue an N(13), A(3), and F(0) (read registers) on the respective Dataway lines. This action corresponds to t_0 in Fig. 2-2 where all signals use negative logic (e.g. 0 V = logic 1). After the F(0) command is issued, the ADC module must gate the contents of its buffer onto the data lines of the Dataway (and remove its LAM signal) within 300 ns. The S_1 signal is issued at t = 400 ns and gates the information into the crate controller (or to the host computer). The S_2 signal, issued at t = 700 ns resets the status lines and signals the end of the Dataway operation (e.g. may re-arm the ADC for the next sample). All data and command signals on the Dataway are TTL level negative logic signals (e.g. logic 1 is < 0.8 V, logic 0 is > 2.0 V).

Three special commands, called "common control signals", operate on all modules simultaneously. These commands are the initialize (Z), clear (C), and inhibit (I) signals. The initialize signal is used after turning on the crate to reset all internal registers to a defined state. This command has priority over all Dataway operations. The clear command is used to reset all I/O registers connected to the Dataway and the inhibit command will halt all activity on the Dataway (without losing information). As a safeguard against loss of information, the initialize and clear commands require the S_2 signal to be present before they will execute.

In closing this chapter it should be noted that this material has only cited the major aspects of the CAMAC Standards. Detailed discussions of the standards are included in the literature of Ref. 3.

CHAPTER III

EQUIPMENT

Sec. 3.1 Introduction

The equipment making up the data acquisition system was chosen for its flexibility and expandability. As mentioned in Ch. I and shown in Fig. 1-1, the major components of the system are the PDP-11/04 computer system, the CAMAC digitizing system, and the Tektronix 4025 graphics terminal. In this chapter, each piece of equipment is discussed with respect to its purpose within the overall system. References are made to the existing documentation for detailed electrical descriptions and options for some of the equipment.

Sec. 3.2 The Host Computer

The main part of the data acquisition system is a Digital Equipment PDP-11/04 minicomputer which is used to control the CAMAC equipment, transfer data to and from disk files, and execute software programs to analyze and display data. The computer was purchased as DEC system #SR20RRA-LA. The components of this system are listed in Table 3-1. The documentation for the 11/04 system includes both operator's instructions for using the system and engineering drawings for trouble shooting and repair. Those intending to operate the system should become familiar with the "Introduction to RT-11", the "RT-11 User's Guide", and the appropriate language reference manual (FORTRAN, BASIC, or ASSEMBLY LANGUAGE).

The front panel of the computer contains a run/halt switch to halt the CPU and an LED "run" indicator. By use of this switch, program

TABLE 3-1

Components of the PDP-11/04 Computer System

The 11/04 central processor unit (CPU)

32 kb of active MOS dynamic RAM

512 kb dual floppy disk drive (RX01)

Bootstrap ROM module (9301-YF)

Parity generator/checker module

DL11-W serial interface board

Unibus terminator (9302)

LA36 printer terminal

RT-11 real time operating system

Documentation kit

execution can be interrupted (e.g. as the Tokamak is being fired,to protect against noise-induced errors) and resumed without loss of information. Through a slight modification of the front panel PC board and two additional switches, the halt/run switch has been made available at a remote location (within the experimental screen room) as well as on the front panel. These modifications and additions are shown in Fig. 3-1. Additional LED's on the front panel and in the screen room indicate which halt switch is enabled.

Sec. 3.3 The DL11 Serial Interface

Communication between the computer and the user is via the DL11-W serial interface board which plugs into the backplane of the computer. The DL11 is a buffered communication interface which allows for the exchange of parallel information from the computer with serial information from the terminal. Also contained on this board is a real time clock running at the AC line frequency. This clock can be set by the RT-11 monitor and read by computer programs with a minimum time interval of 16.6 ms (60 Hz). Through microswitches on the PC board, the DL11 can be configured to be compatible with different types of terminals. Options include separate transmit and receive data transfer rates (known as "baud"), odd or even parity, and variable word length. The DL11 can be configured as either an RS-232 interface or a 20 mA loop interface.

Two terminals are presently available in the data acquisition system, the LA36 printer terminal and the Tektronix 4025 graphics terminal. The LA36 is a 20 mA loop device while the 4025 is an RS-232 device. To simplify changing from one terminal to the other, the option switches on the DL11 (which resides inside the computer cabinet) have been extended,

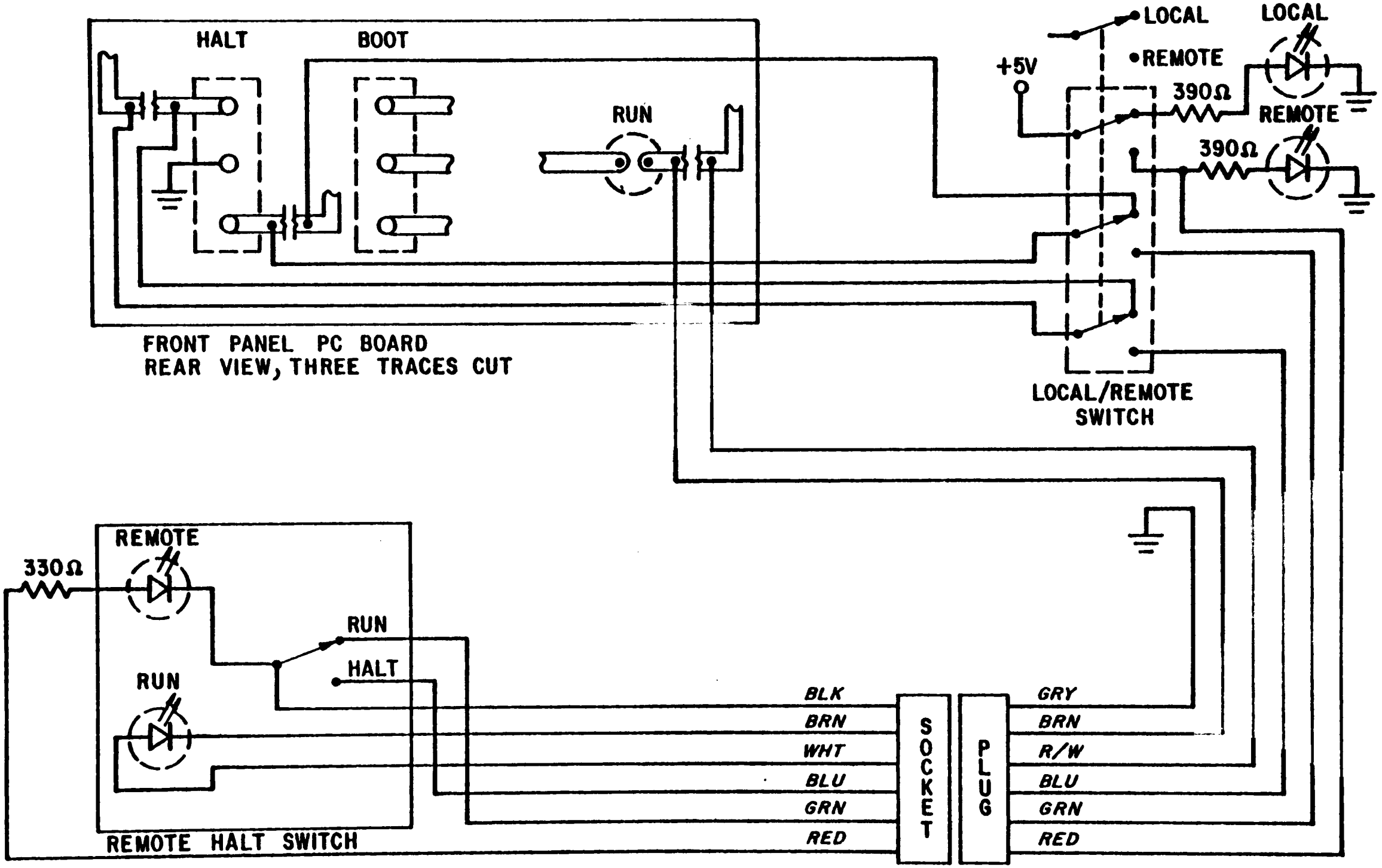


Fig. 3-1. Remote Halt/Continue Switch and PC Board Modifications

via a 40 conductor ribbon cable, to a front panel above the computer. This front panel switch board is shown in Fig. 3-2 and a schematic is given in Figs. 3-3, 4. The switch panel allows the user to select one of three possible terminals by turning a single three-position switch. Position 1 is wired for 20 mA loop operation and is dedicated to the LA36. Position 2 is wired for RS-232 operation and is dedicated to the 4025. Position 3 is wired for RS-232 operation but is not dedicated to any one terminal. The transmit and receive rates (and other options for position 3) are selected independently for each terminal by miniature switch packs on the front panel. All other options for positions 1 and 2 have been hard-wired.

Only one terminal can be transmitting to the computer at any given time but all three may listen by setting their "echo" switches to the "ON" position. Terminals 1 and 2 each have two sockets connected in parallel; one on the back of the switch panel and one in the experimental screen room to allow for remote operation. However, only one terminal should be installed for each position.

Sec. 3.4 Optical Isolation

Typically, the 4025 terminal is used as a remote terminal at the experiment. To protect the terminal against large induced voltages, an optical isolation circuit has been designed and installed in the RS-232 line between the terminal and the switch panel. A schematic of the circuit is shown in Fig. 3-5. The isolator consists of two identical but electrically isolated circuits,each containing a transmitter, a receiver, and a power supply. The transmitter in each side drives a

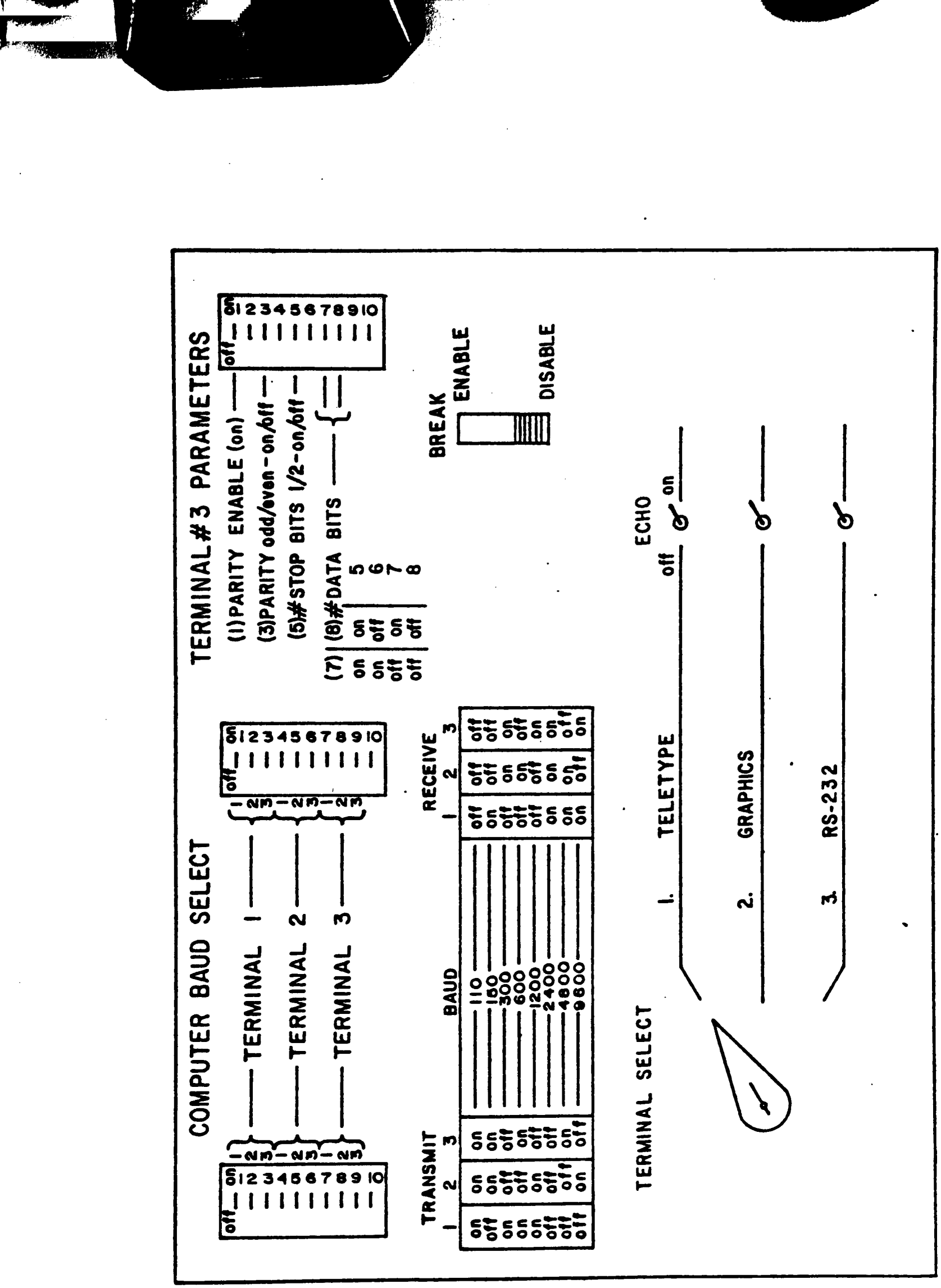


Fig. 3-2. Front Panel of Terminal Selection Board

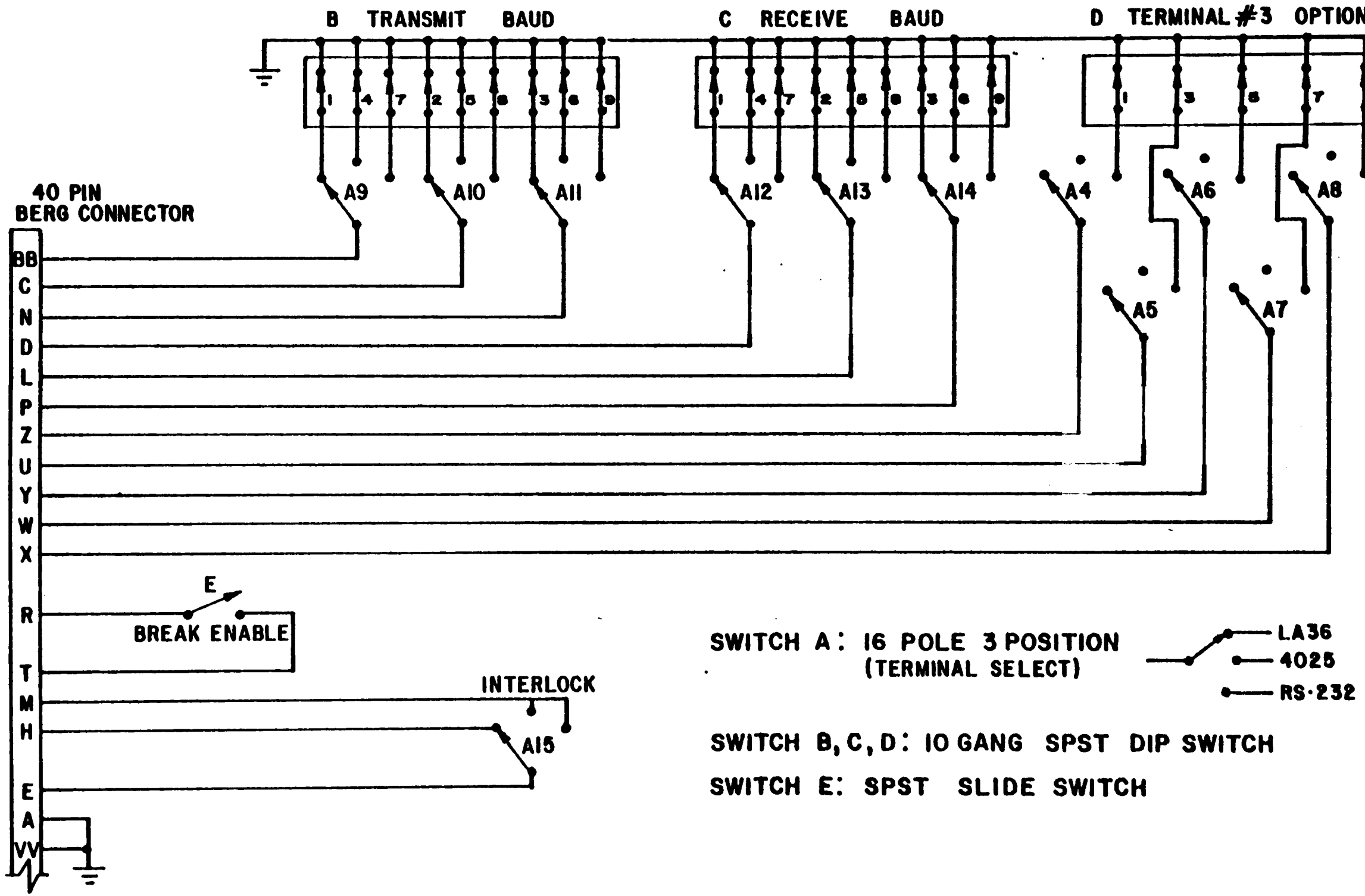


Fig. 3-3. Schematic of Terminal Selection Circuit

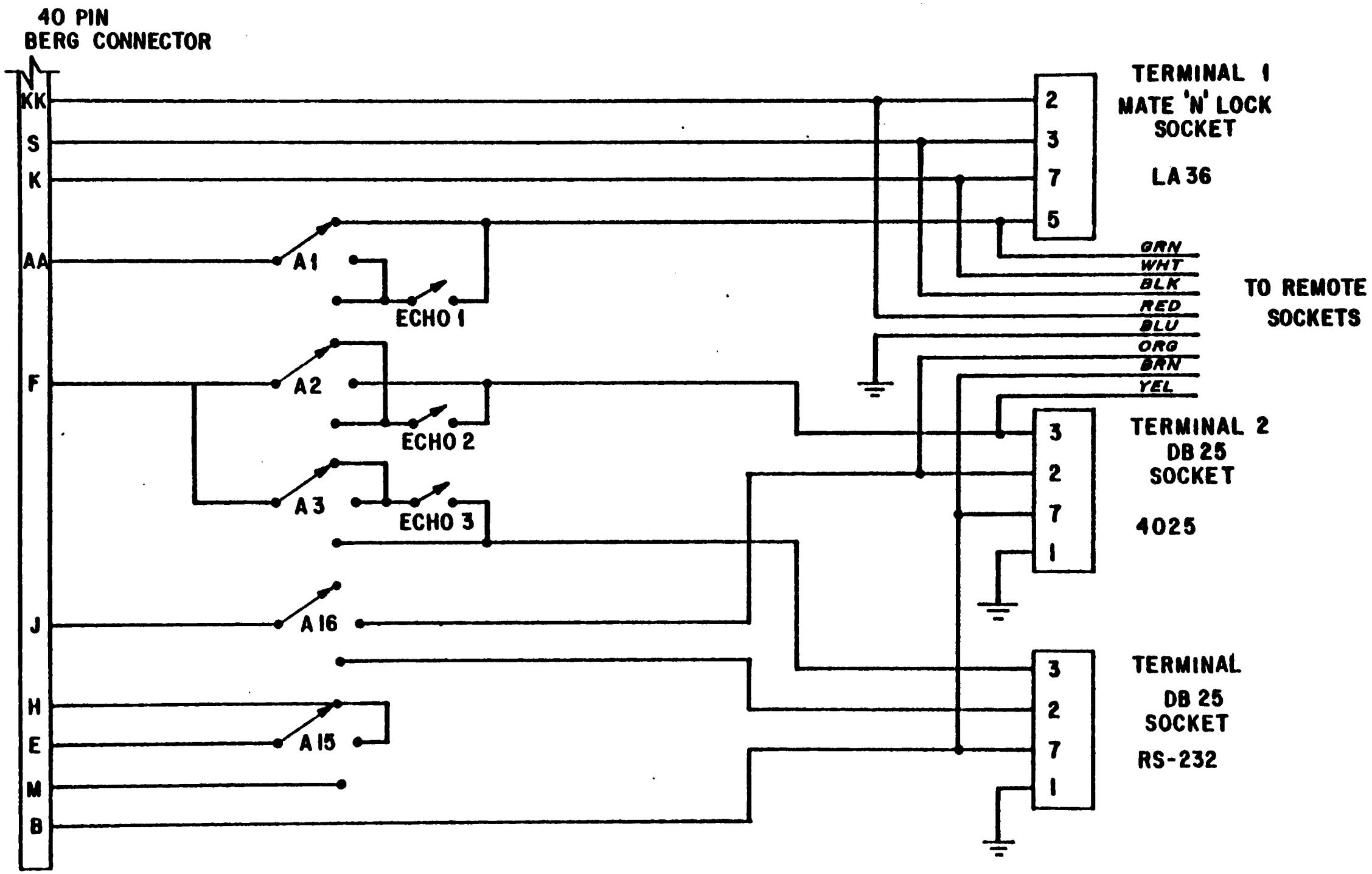


Fig. 3-4. Continuation of Fig. 3-3

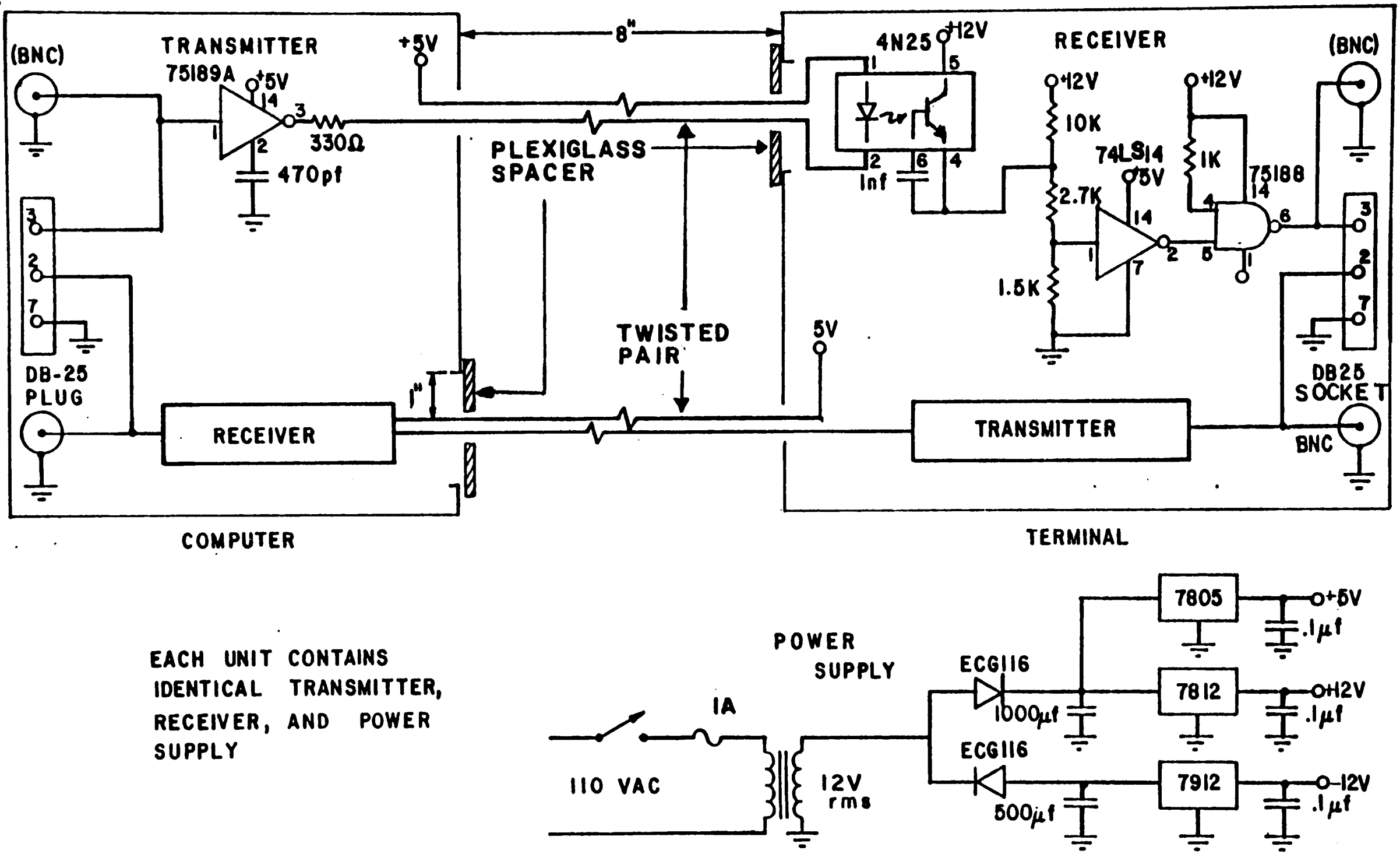


Fig. 3-5. Optical Isolation Circuit

4N25 opto-isolation IC in the other side. This method provides for at least 2.5 kV isolation.

Sec. 3.5 The Communications Interface

An interface has been designed to selectively connect between two terminals and two computers. This unit also serves to monitor any activity on the communication lines. The original purpose of the interface was to allow the 11/04 to communicate with the central campus computer (via modem and timeshare lines) but the unit has been useful as a general purpose monitor. The front panel of the interface is shown in Fig. 3-6 and a schematic is given in Fig. 3-7. The unit is typically put in series with the 4025 terminal but can be used in any RS-232 serial link.

Through switches on the front panel of the interface the operator can connect either terminal to either computer or have the two computers talking to each other. Three LED's on the front panel monitor activity on various data lines and can be used to debug problems in the communication link.

Sec. 3.6 Terminals

The LA36 is a general purpose terminal using a standard ASCII keyboard for input and a 30 character/second impact printer for output. Input and output transmission rates (baud) must be the same and are switch selectable for 110, 150, and 300 baud. Detailed information and maintenance procedures for this terminal can be found in the LA36 User's Manual.

The Tektronix 4025 is a microprocessor-based graphics terminal using an ASCII keyboard for input and a CRT for output. The 4025 is

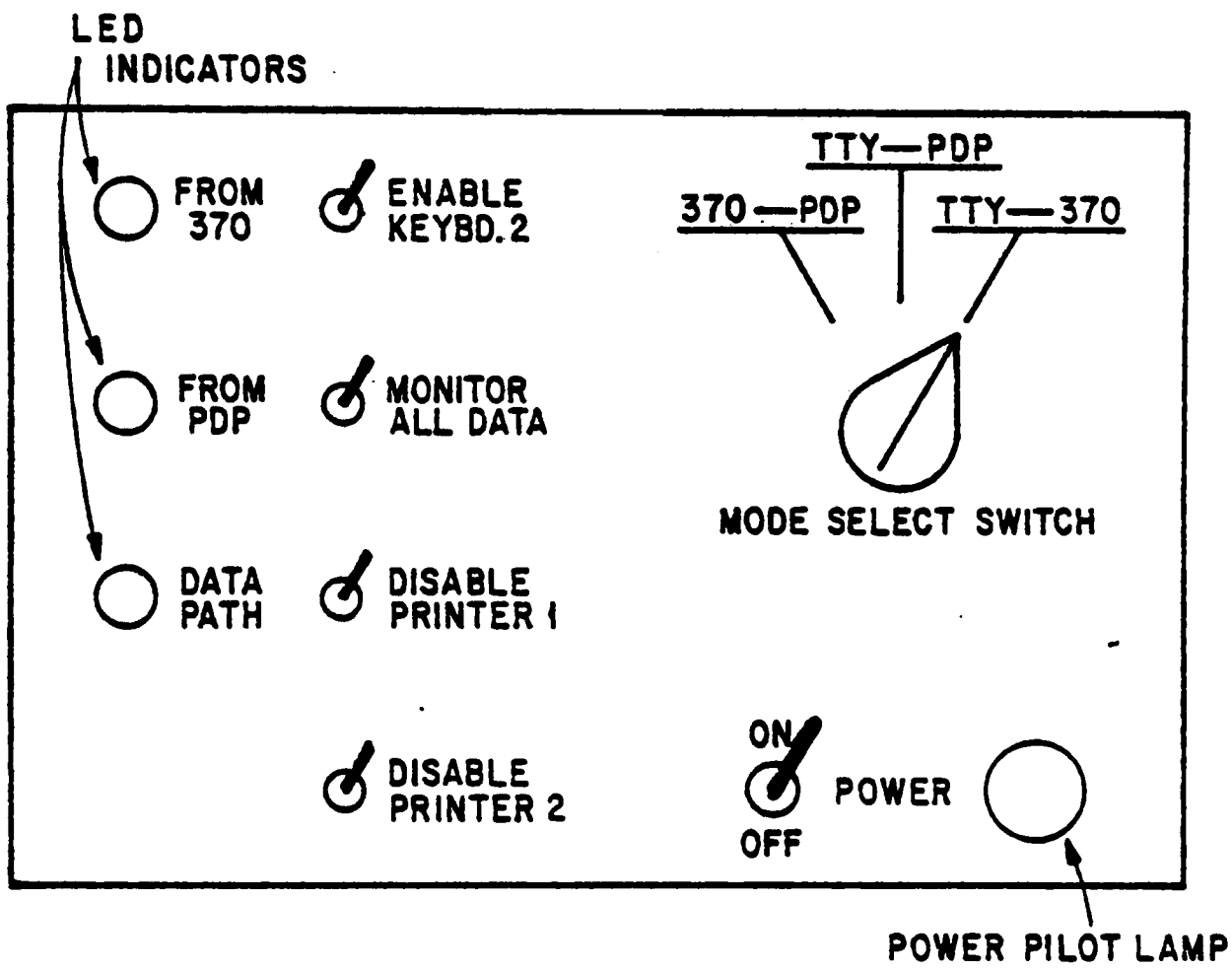


Fig. 3-6. Front Panel of Communications Interface

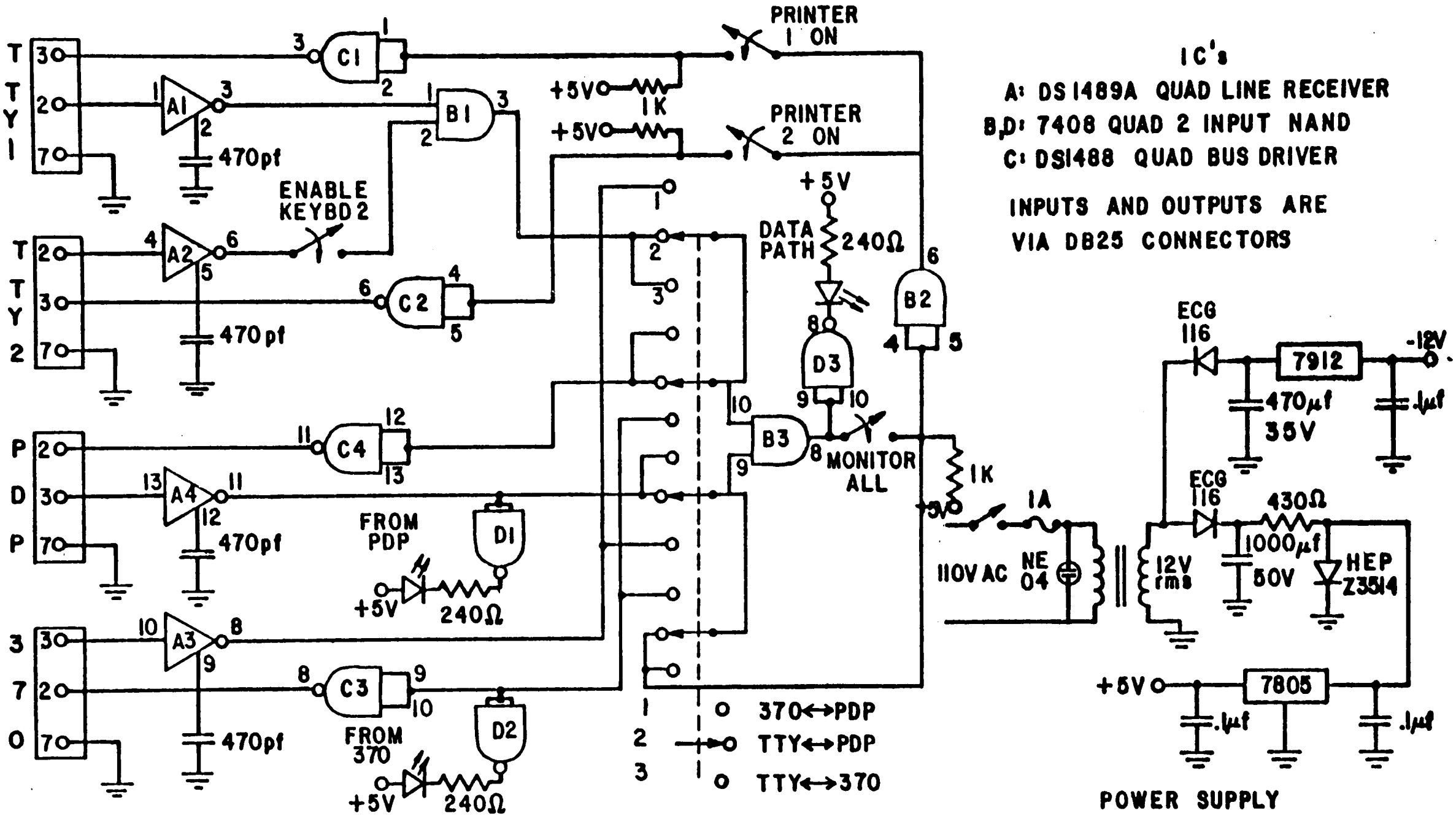


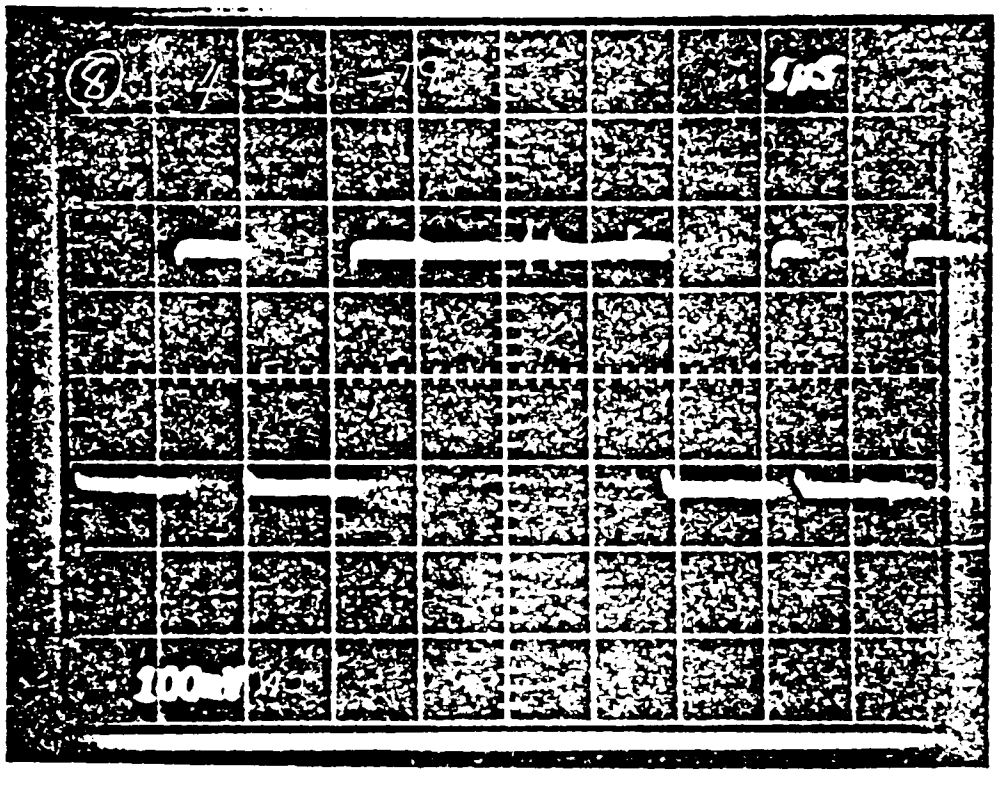
Fig. 3-7. Schematic Diagram of Communication Interface

capable of dividing the screen into upper and lower areas called the "workspace" and the "monitor", respectively. Text from the keyboard or computer can be directed to either area. Local memory allows the 4025 to store up to ten pages (screens) of text which can then be "scrolled" on or off the screen using keyboard commands. Other options and features of the 4025 are discussed in the 4025 User's Manual and the 4025 Programmer's Guide.

Sec. 3.7 11/04 - CAMAC Interface

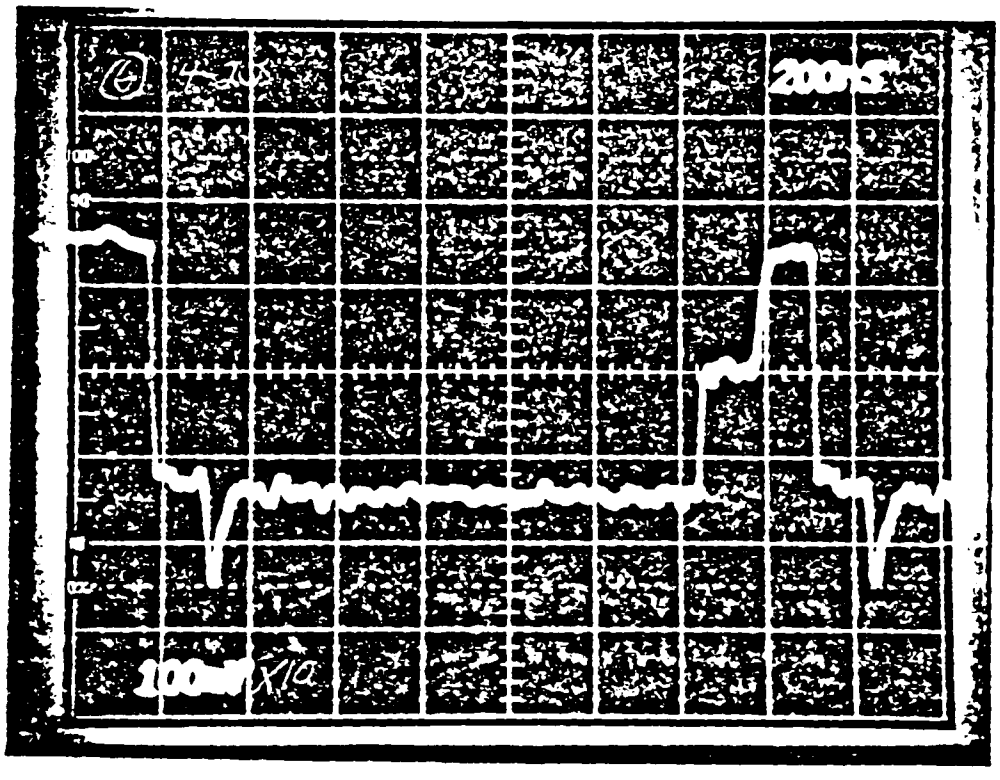
The circuit boards which make up the 11/04 system are interconnected by a common backplane called the Unibus.[5] The Unibus consists of 56 signal lines which carry data and commands to each board within the computer. For making connections to external devices, the computer backplane is brought outside the cabinet via an Unibus extension cable. A 14.2 m Unibus extension cable has been installed between the computer and the CAMAC crate. Tests were made to ensure that the capacitive loading of the cable did not degrade the rise and fall times of the signals on the bus. The results are shown in the oscillographs of Figs. 3-8 and 3-9. Shot 3-8(a) depicts the signals on an address line without the extension cable. Shot 3-8(b) shows the effect of the unterminated extension cable. Shot 3-9(a) and (b) show the response on the bus with the Unibus terminator board properly installed in the connector block at the CAMAC crate. The rise and fall times to TTL levels (0.8 V - 2.0 V) are about 20 ns and are almost identical to the response of the unextended backplane.

To protect the system from electromagnetic interference (EMI) the Unibus extension cable and the remote terminal lines are enclosed in a



Vertical
Scale
1 V/cm

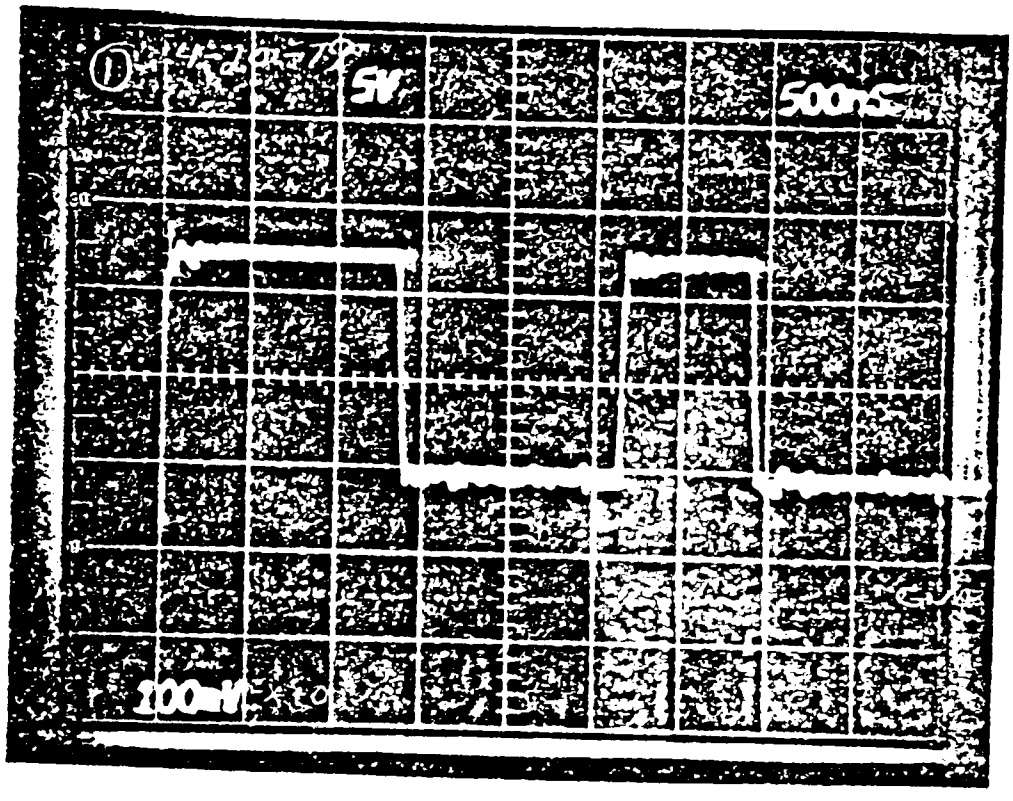
(a) Standard Bus, No Extension



Vertical
Scale
1 V/cm

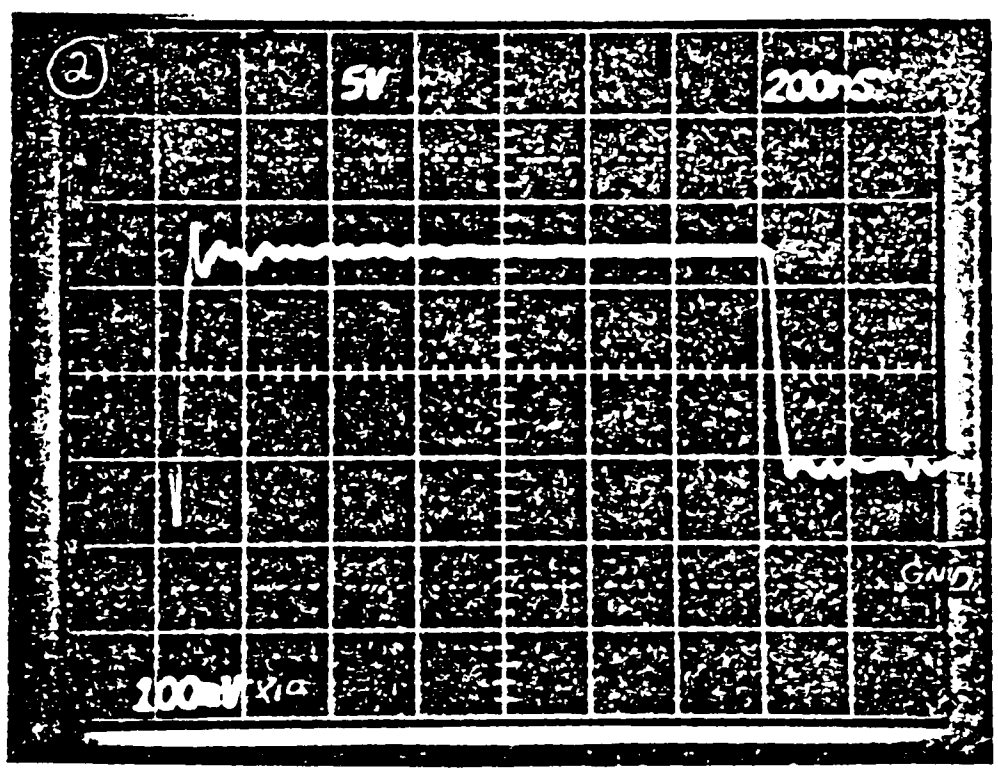
(b) Extended Bus, Unterminated

Fig. 3-8. Signals on the Unibus



Vertical
Scale
1 V/cm

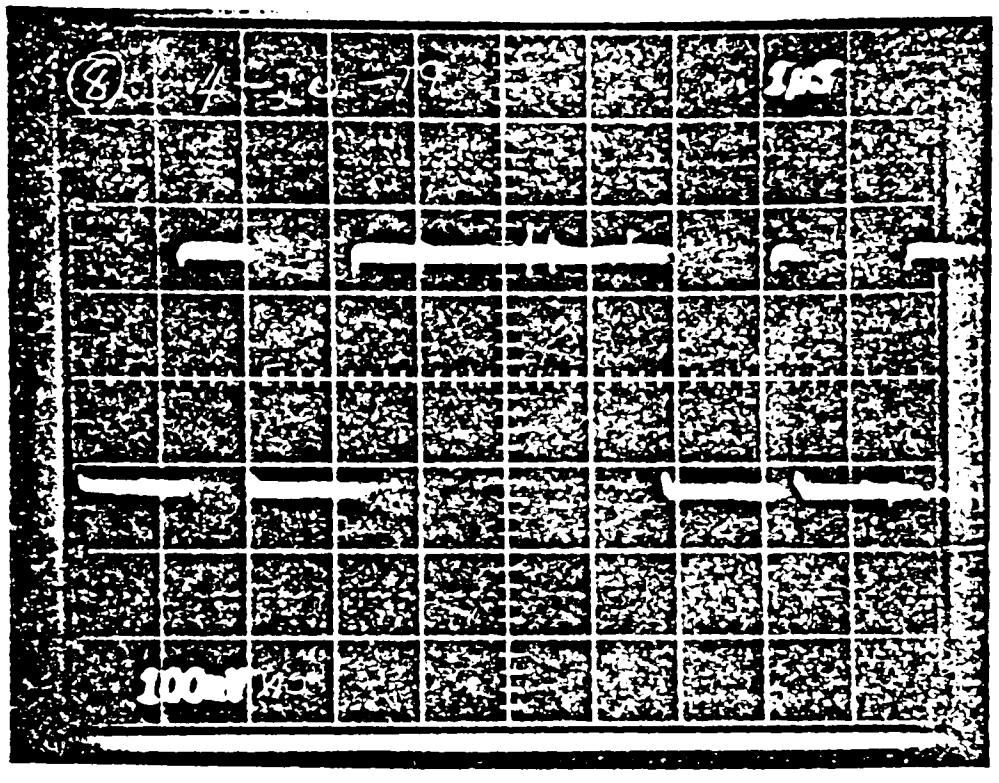
(a) Bus Extended and Terminated



Vertical
Scale
1 V/cm

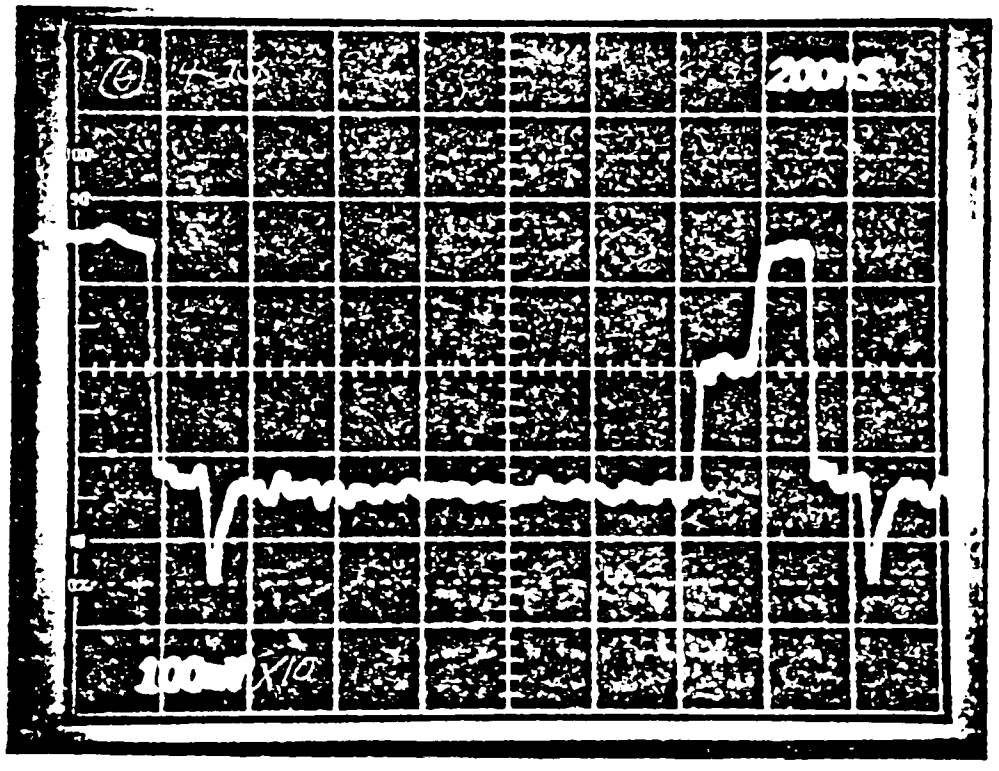
(b) Expanded Time Scale of (a)

Fig. 3-9. Signals on the Unibus



Vertical
Scale
1 V/cm

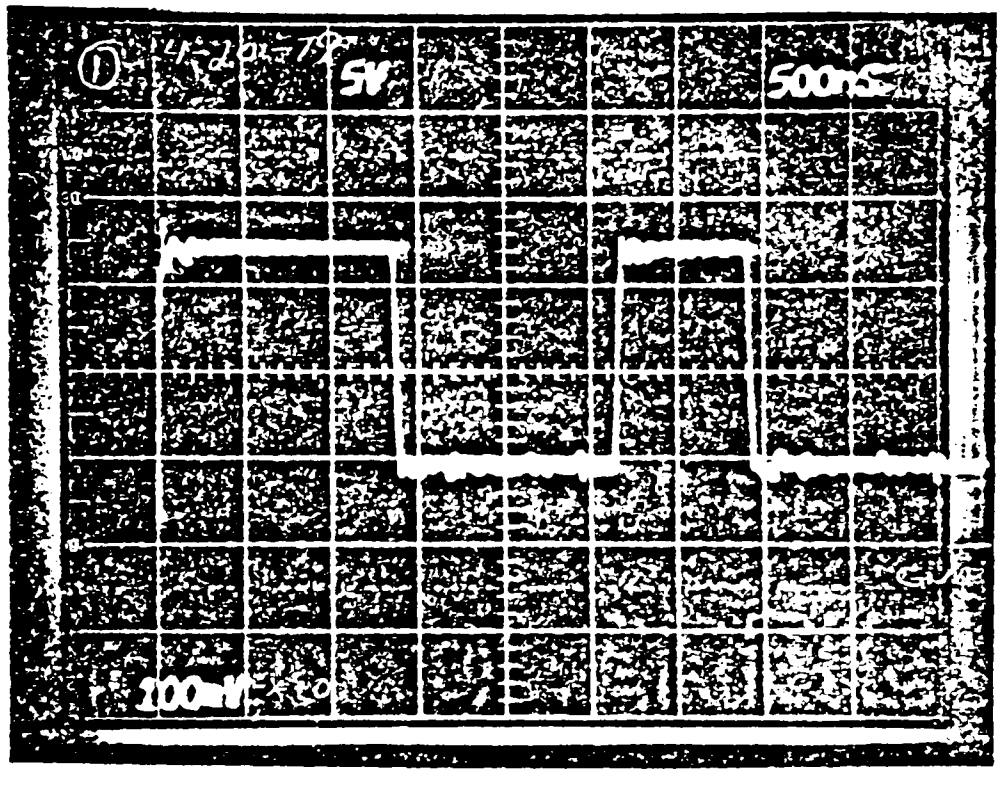
(a) Standard Bus, No Extension



Vertical
Scale
1 V/cm

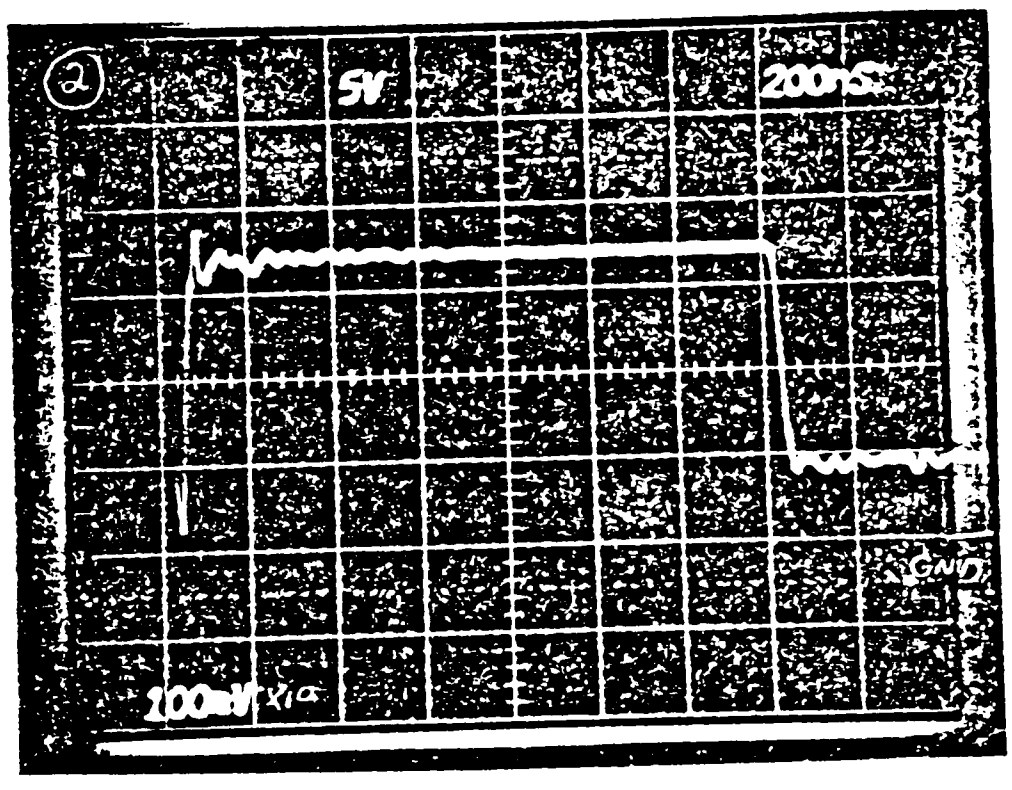
(b) Extended Bus, Unterminated

Fig. 3-8. Signals on the Unibus



Vertical
Scale
1 V/cm

(a) Bus Extended and Terminated



Vertical
Scale
1 V/cm

(b) Expanded Time Scale of (a)

Fig. 3-9. Signals on the Unibus

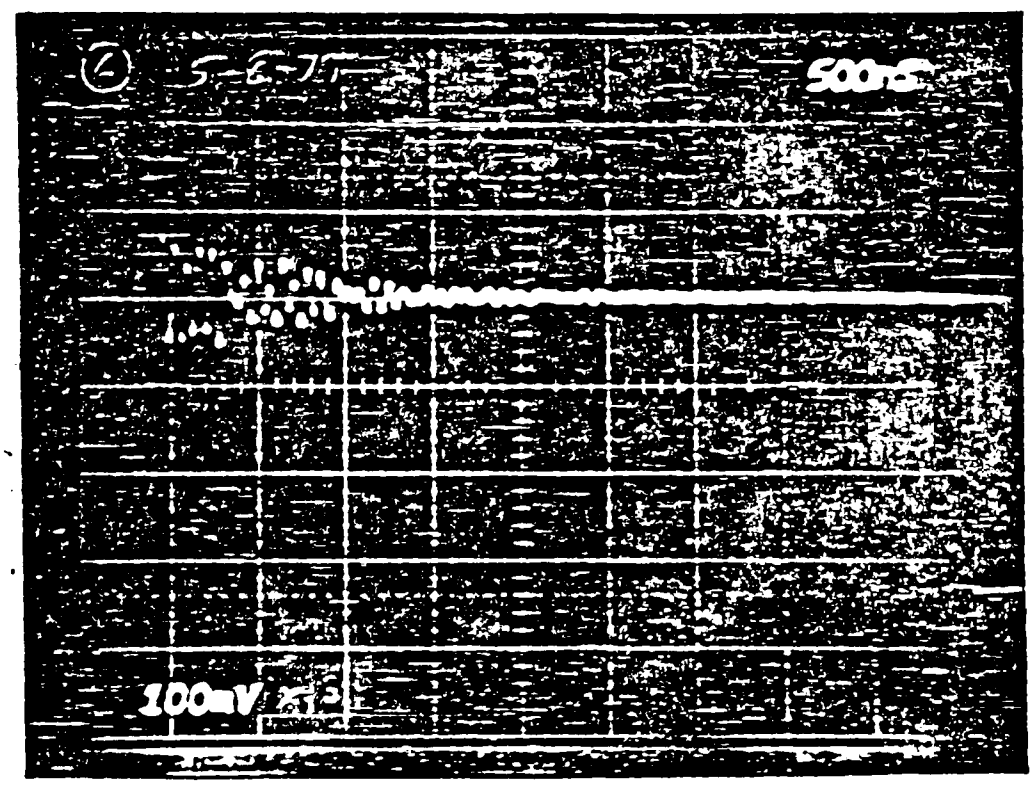
continous conduit connecting the computer cabinet to the experimental screen room where the CAMAC crate resides. The conduit is approximately 11 m long and is constructed from .3 m length of .102 m O.D. EMT electrical conduit. Noise on the Unibus was measured while the Tokamak was fired. The results are shown in Fig. 3-10. The peak noise voltage was found to be about 0.75 V, well below the 1.5 V noise margin of the Unibus.

Sec. 3.8 The DCC-11 Interface

The Unibus extension cable has effectively brought the resources of the 11/04 to the experimental screen room. As discussed in Sec. 2.3 the CAMAC modules are interconnected by the Dataway which is not directly compatible with the Unibus. A Standard Engineering model DCC-11 crate controller is used to translate data and command signals between the Dataway and the Unibus. A detailed explanation of this interface can be found in the DCC-11 User's Manual. Briefly, the DCC-11 is a "dumb" crate controller that maps the CAMAC station addresses and subaddresses onto a block of 1024 Unibus addresses. Each address and subaddress (and the data they contain) can be uniquely accessed by the 11/04 instruction set.

Sec. 3.9 CAMAC Modules

Data from the experiment are transfered to the computer through CAMAC modules. Presently incorporated in the sustem are two 8 channel waveform digitizers for time resolved data and a 32 channel scanning DVM for single sample data. Nine analog buffers have been built (a 3 channel prototype and a 6 channel final version) to amplify and filter



Vertical
Scale
1 V/cm

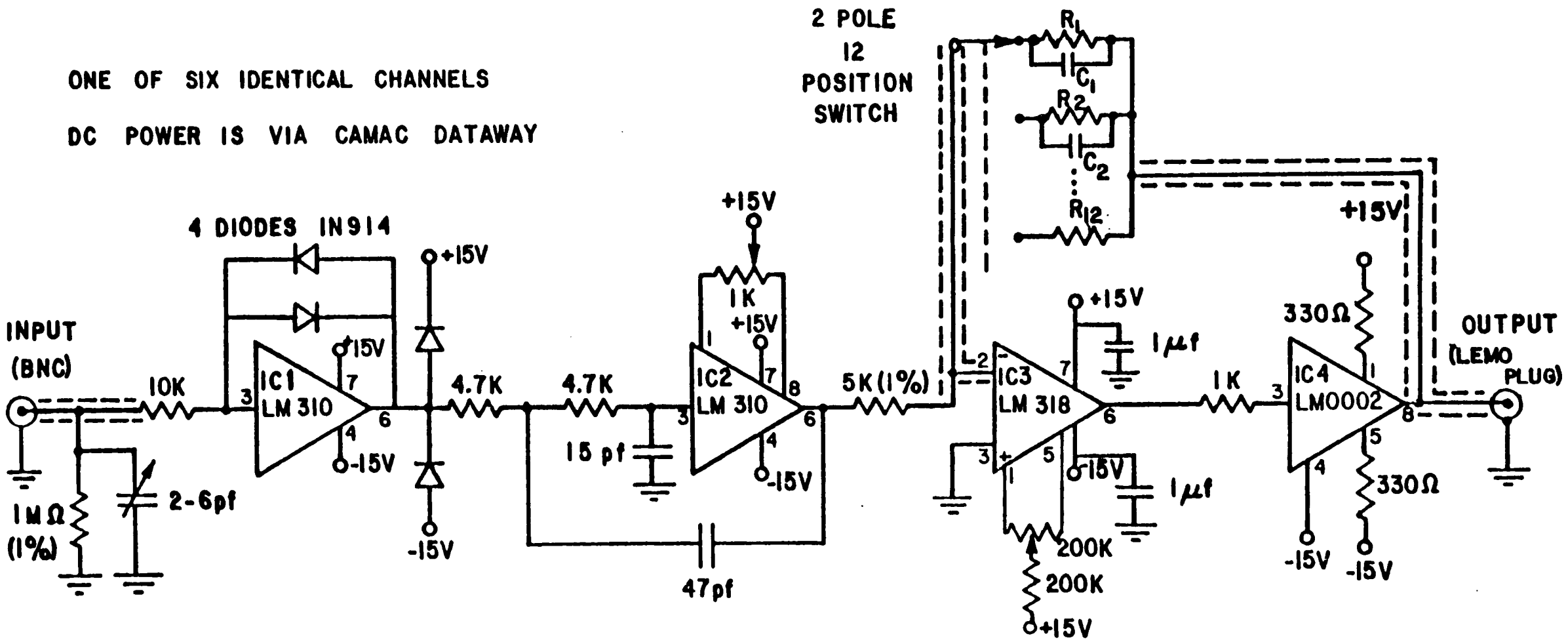
Fig. 3-10. EMI Induced on Unibus During Tokamak Discharge

the signals (if necessary) before digitizing. A 2 channel driver has also been built to display data on a Tektronix 604 monitor.

The two 8 channel digitizers are LeCroy model 2264, each occupying three CAMAC stations. The model 2264 consists of a single ADC which is multiplexed over 1 to 8 inputs. The maximum sampling rate is 4 MHz for a single channel and 500 kHz for 8 channels. Each of these digitizers requires a LeCroy model 8800/8 buffer memory module for storing the digitized data. Front panel controls allow the user to select the sample rate and the number of channels to sample. These units also have the ability to record pre-trigger data as well as post-trigger data. The front panel switch positions can be read (but not set) by the controller.

The 9 buffer amplifiers have been built to provide a 1 MΩ input impedance for the diagnostics and 50 Ω drive capability to accommodate the 8 channel digitizers. The buffers also provide input over-voltage protection, low pass filtering, and switch selectable gain. The 6 channel version has 12 gain settings from X0.01 to X100 and includes a logic circuit which allows the controller to read the setting of the gain switch for each channel. The schematic for a typical buffer channel is shown in Fig. 3-11 and the position readout logic is shown in Fig. 3-12.

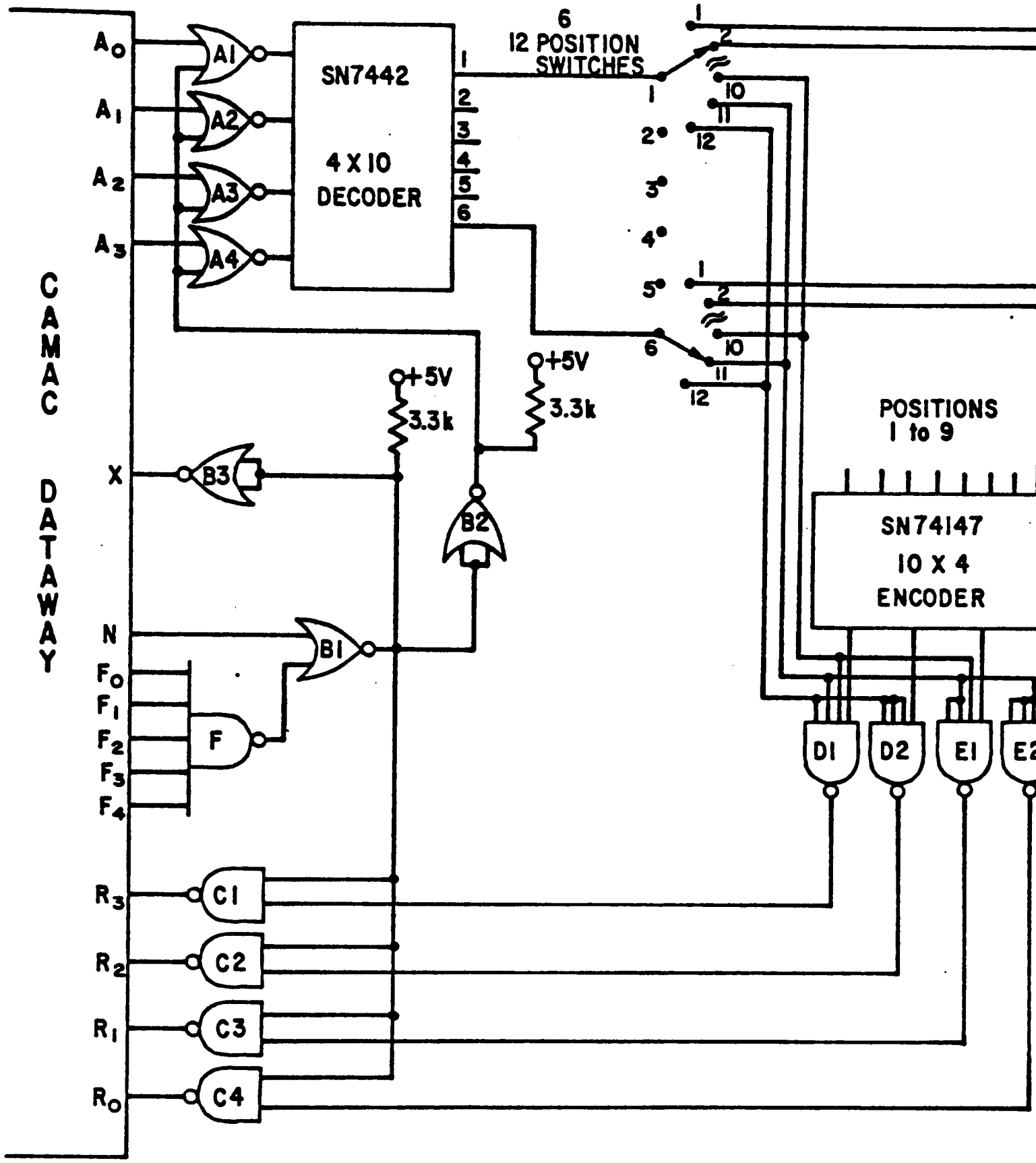
The 8 channel digitizers also include a DAC circuit to reconstruct the data stored in the buffer memory for display on an oscilloscope. A 2 channel driver has been built to display this reconstructed data on a Tektronix 604 monitor. The driver includes vertical position control, X1 or X10 vertical gain, and a choice of continuous or single sweep.



R-C PAIRS (RESISTORS 1%)

POS	R(KΩ)	C(pf)	GAIN	POS	R(KΩ)	C(pf)	GAIN
1	.125	820	.025	7	12.5	8	2.5
2	.25	390	.05	8	25.0	3.9	5.
3	.5	200	.1	9	50.0	2.2	10.
4	1.25	82	.25	10	125.0	–	25.
5	2.5	39	.5	11	250.0	–	50.
6	5.0	22	1.0	12	500.0	–	100.

Fig. 3-11. Schematic of 6 Channel Analog Buffer



I.C.'S

A: QUAD 2 INPUT NOR 7402
B: QUAD 2 INPUT OPEN COLLECTOR NOR 74LS33
C: QUAD 2 INPUT OPEN COLLECTOR NAND 7403
D,E: DUAL 4 INPUT NAND 7420
F: 8 INPUT NAND 7430

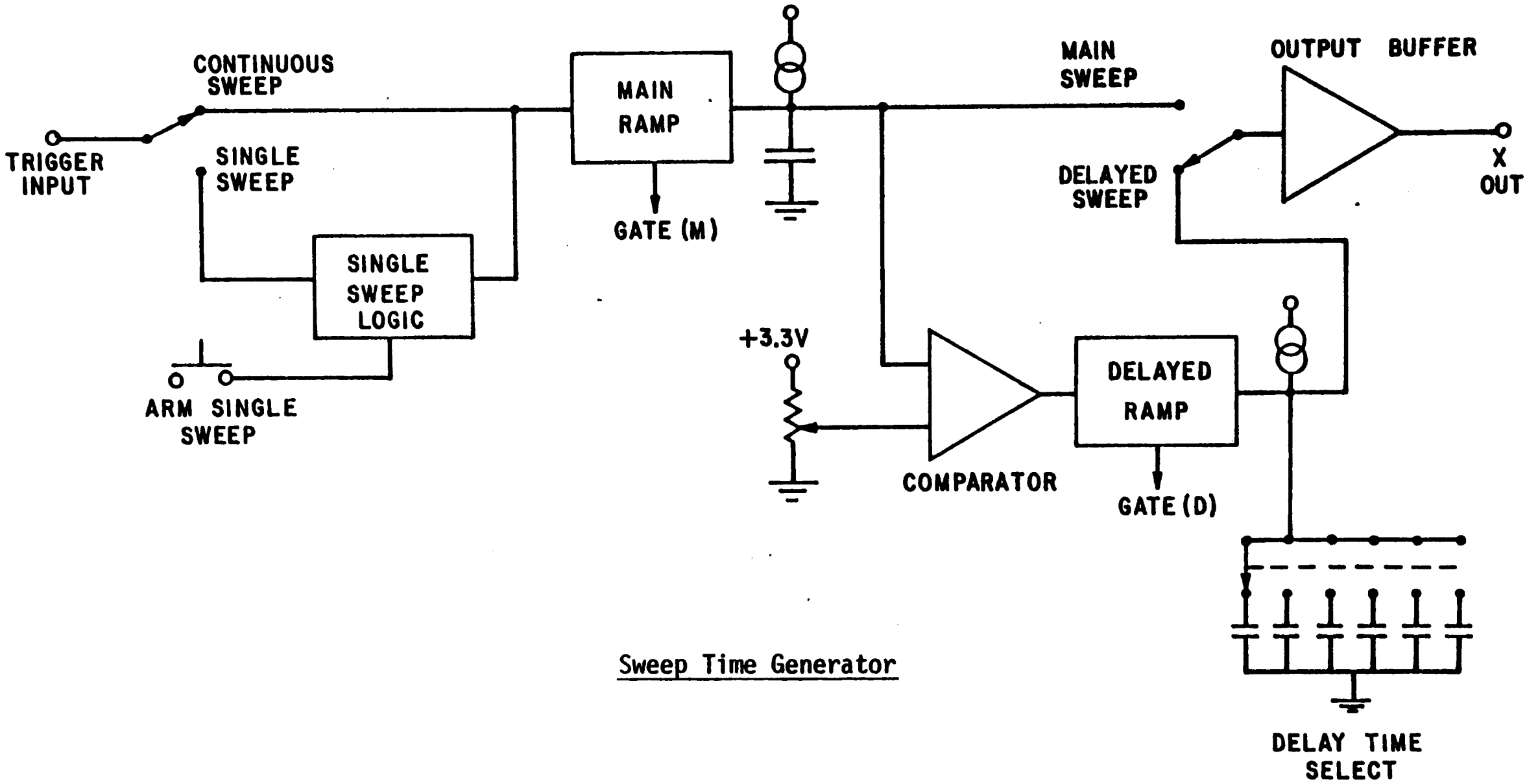
D.C. POWER FROM CAMAC DATAWAY

Fig. 3-12. Switch Position Readout Logic for 6 Channel Analog Buffer

Also, variable sweep time and a delayed trigger feature allows any portion of the waveform to be expanded in time. A block diagram of the display driver is shown in Figs. 3-13 and 3-14.

The final module in the present system is a LeCroy model 2232A, 32 channel, scanning DVM. This unit contains a single ADC which is multiplexed over the 32 inputs. The voltage at each input is converted to a 12 bit binary word and stored in a local 32 word X 12 bit buffer memory.

With the present configuration only 5 of the 25 CAMAC stations are free for future expansion. Many of the modules, however, draw only their DC power from the crate and make no connection to the Dataway. These modules could reside outside the crate which would then leave 13 stations free for future use.



Sweep Time Generator

Fig. 3-13. Block Diagram of 2 Channel Display Driver

INTENSITY CONTROL

NOTES:

(1) from fast level shift detector

(2) true when "Main Intensity Delayed" switch is in the "Intensify" position

(3) true if trigger is not the first to occur during a sweep

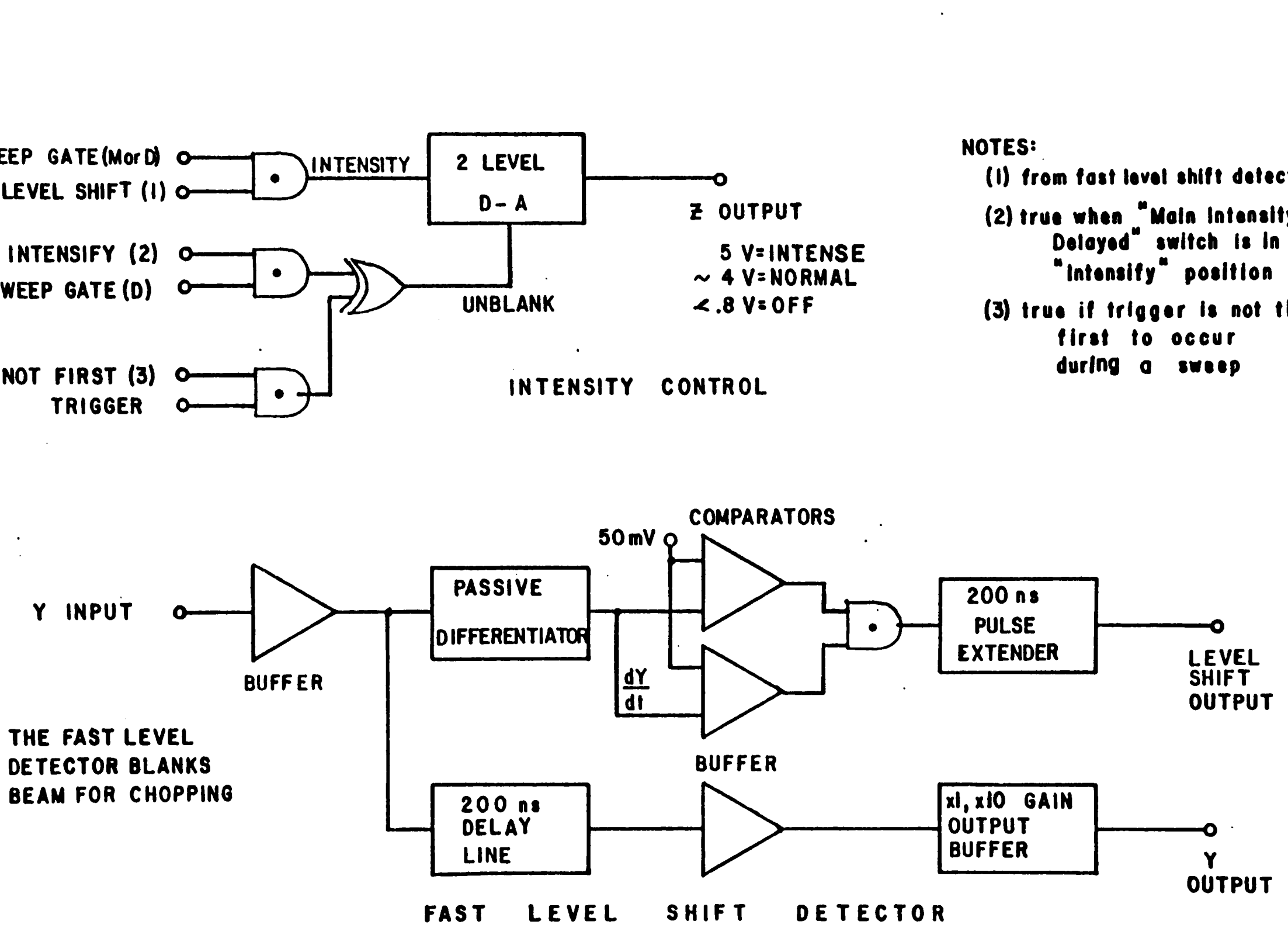


Fig. 3-14. Continuation of Fig. 3-13

CHAPTER IV

SOFTWARE

Sec. 4.1 Introduction

A set of BASIC subroutines has been written to control the hardware described in Ch. III. These subroutines have been combined with data manipulation and error checking routines to form a versatile, expandable, data storage program. The program is structured as a command interpreter and address table. Commands are executed by subroutines, the starting addresses of which are stored in the address table within the main program. When a command is entered from the keyboard, the program searches the address table for the start address of the corresponding subroutine, executes the subroutine, and returns to the main program to wait for the next command. The subroutine, when necessary, may request additional information from the operator while it is executing. As new equipment is acquired, new command subroutines can be added anywhere in the main program and their addresses added to the address table. This chapter will discuss the structure and flow of the software to allow future users to make additions without re-writing the entire program, while Ch. V is a set of instructions for loading and running the storage program.

Sec. 4.2 System Information

Some knowledge of the 11/04 system structure is required in order to understand the storing routines. Those points that pertain to the routines are discussed here. For more complete information the reader is referred to the 11/04 documentation kit.

The storage routines are written to run on the 11/04 system (with the equipment listed in Table 3-1) and a Tektronix 4025 graphics terminal. The RX01 disk drive is a file structured device. Files on a disk can be created and manipulated by the RT-11 monitor and by an executing program. A "directory" file, installed on each disk, contains the title, length, creation date, and position (on the disk) of all files on that disk. A program stored on a disk must be downloaded into active memory before it can be executed. Loading a program does not alter the file stored on disk.

Space on a floppy disk is allocated in "blocks" where a block is 512 bytes. A standard disk, after being "initialized", will contain 480 free blocks. The RT-11 "initialize" command prepares a disk for use in the 11/04 system by creating a directory (or erasing the existing directory if the disk has been used before). The two disk drives are labeled DX0 and DX1 where "DX" indicates an RX01 drive. In the default configuration, DX0 is the system disk containing the RT-11 monitor programs and device handlers while DX1 is used for general purpose storage.

A directory of the disks used with the storage routines is shown in Table 4-1. Each entry in the directory includes the title, the length (in blocks), and the creation date (or last modification date). The system disk (in DX0) contains the required RT-11 monitor programs and the data storage routines. The disk in drive DX1, as shown in Table 4-1, is a specially formatted disk for storing data files. Each of the files on this disk has a specific purpose that will be discussed in later sections.

File titles consist of two parts separated by a period. The first part is a 1-6 character name usually chosen to describe the contents or function of the file. The second part is a 1-3 character "type". The

TABLE 4-1

Directory of Storage Routine System Disk and Formatted Data Disk

```
 02-Jan-80
CATDIR.BAS      3 01-Nov-79    184      0      .SAV     40 20-Sep-79    206
CATDIR.COM      1 06-Nov-79    103      OVRNEW.BAS      25 08-Nov-79    370
CLRLOW.BAS      1 12-Oct-79    260      PIP    .SAV     16 11-Mar-78    107
CONFIG.INF      1 18-Oct-79    271      PLT200.BAS       9 01-Nov-79    319
CREATE.BAS      3 17-Oct-79    264      RAD001.FMT       2 18-Oct-79    262
DATA   .HED     8 07-Nov-79    362      RAD002.FMT       1 23-Oct-79    275
DIR    .SAV    17 11-Mar-78    144      RAD003.FMT       2 24-Oct-79    273
DUMMY  .BAS     1 08-Nov-79    193      RAD005.FMT       2 07-Nov-79    276
DUP    .SAV    21 24-Mar-78    123      RAD006.FMT       2 07-Nov-79    278
DXMNSJ.SYS     63 11-Mar-78     38      READ2  .BAS       2 24-Oct-79    280
EDBAS  .LST     2 22-Mav-79    180      RED200.BAS      10 24-Oct-79    299
EDIT   .SAV    19 11-Mar-78    161      RT11ED.LST       2 13-Jul-79    182
FIXRED.BAS      3 24-Oct-79    309      STARTS.COM       1 17-Oct-79    250
FORMAT.FMT      1 24-Oct-79    284      STO200.BAS      14 08-Nov-79    423
GPKEYS.LST      1 05-Sep-79    191      STRIP  .BAS       1 02-Oct-79    267
GRA200.BAS      2 01-Nov-79    328      SWAP   .SYS      24 11-Mar-78     14
LENGTH.BAS      1 15-Oct-79    261      TITLE  .REC       1 03-Oct-79    247
LODREC.BAS      2 24-Oct-79    282      TITLE  .TXT       1 25-Oct-79    270
MACHIN.DAT      7 07-Nov-79    330      TT     .SYS       2 11-Mar-78    101
NL     .SYS     2 11-Mar-78    105
 39 Files, 316 Blocks
 164 Free blocks
```

System Disk DX0

```
 02-Jan-80
TITLE .TXT      1 19-Dec-79     14      < UNUSED >      3                15
INITIA.LIZ      1 19-Dec-79     18      CATALO.G        2 19-Dec-79      19
< UNUSED >    473               21
 3 Files, 4 Blocks
 476 Free blocks
```

Data Disk DX1

Each entry includes: the title and type, the length, the creation or last modification date, and the position on the disk. In this example, no data files had been recorded.

file type can be chosen arbitrarily but the RT-11 monitor is programmed to recognize certain default types.[6]

Files on the system disk in Table 4-1 fall into three categories:

1) Those files required by the RT-11 monitor

2) Those files required to run the storage routines

3) Often used support programs

Files in the first group include:

SWAP.SAV TT.SYS DXMNSJ.SYS PIP.SAV

DUP.SAV DIR.SAV STARTS.COM TITLE.TXT

Files in the second group include:

0.SAV CONFIG.INF FORMAT.FMT MACHIN.DAT

STO200.BAS GRA200.BAS PLT200.BAS OVRNEW.BAS

FIXRED.BAS RED200.BAS DATA.HED DUMMY.BAS

CLRLOW.BAS

The rest of the files on DX0 are useful support programs which are kept on the system disk for convenience but could be removed if more room were needed on the system disk.

Sec. 4.3 System Programs

All of the files in group 1 with the exception of TITLE.TXT are RT-11 system files and are discussed in the RT-11 documentation. The use of the TITLE.TXT file is a local convention which was introduced to aid in disk identification. The TITLE.TXT file is a one block text file containing the disk name and number (if applicable) and a brief description of the contents or purpose of the disk. By opening and reading this file, a computer program or remote operator can determine what disks are in each drive.

Sec. 4.4 Data Disks

Before a blank disk can be used, it must be formatted by the RT-11 "initialize" command. If a disk is going to be used as a data disk for the storage routines it must also be formatted by the "CREATE.BAS" program which creates the files shown in Table 4-1. The title of a data disk will always have the form:

DAxxx (e.g. DA006)

where "xxx" is a three digit sequence number. The title file will also contain the initialization date and any comment (up to 450 characters) the operator wished to make. The disk title and initialization date should also be put on the protective cover of the disk.

When a file is copied or edited, it may be moved to a new position on the disk. Since the "INITIA.LIZ" and "CATALO.G" files are accessed quite often, a 3-block blank space has been left near the top of the data disk for these files to move into to prevent them from mixing with the data files and leaving behind holes of wasted disk space. Spaces almost as large as a data file can be wasted in this way. A listing of the "CREATE.BAS" program and a line by line discussion is included in Appendix A.

The "INITIA.LIZ" file on the data disk contains the date and shot number of the last data file stored on that disk, so that shots can be automatically numbered in the correct sequence, even if the computer is turned off (or crashes) during an experiment. Because "INITIA.LIZ" is updated only after each shot has been stored, it is possible that the computer may stop (crash) after a valid data file is stored, but before "INITIA.LIZ" is updated. To minimize the chance of losing a data file this way, the software asks the operator to verify the shot number if the program is restarted.

The "CATALO.G" file on the data disk will contain a directory of all the data files on that disk. This file is loaded at the end of the day by a two step process. First, in RT-11, the "CATDIR.COM" command file is executed to extract the directory information from the data disk and deposit it in a temporary file on the system disk. Then in BASIC, the program "CATDIR.BAS" is executed to transfer the directory information into the catalog file on the data disk.

The file name for each data file is created by the storage routine which combines the date and the shot number, as follows:

File name = YYMMDD.SHT#

Where: YY = the last two digits of the year

MM = the number of the month (1-12)

DD = the day of the month (1-31)

Zeros are added if necessary so that the file name always has 9 characters.

Sec. 4.5 The BASIC Language

The file "O.SAV" on the system disk is the BASIC interpreter and must be loaded into active memory before a BASIC program can be run. The original version was DEC BASIC/RT-11 V01 B. Some new commands have been linked[7] into BASIC to allow for CAMAC operations. These commands are in the form of callable functions and include:

"PEEK" to examine a word in memory

"POKE" to modify a word in memory

"GETB" to examine a byte in memory

"STOB" to modify a byte in memory

"FREM" to determine the amount and location of free memory

"EXEC" to execute a machine language subroutine loaded into

memory by the "POKE" function

"FAST" a special purpose machine language routine for reading data from the LeCroy 2264 digitizers

The memory accessing functions (PEEK, POKE, GETB, and STOB) are required by the DCC-11 controller in order to execute "F" commands and common control commands (as discussed in Sec. 2.5). The method by which the commands are executed is discussed in the DCC-11 User's Manual. The "FAST" routine simplifies the reading of data and status information from the waveform digitizers. The "FREM", "EXEC", and "POKE" functions allow machine language routines to be loaded and executed by a BASIC program. Listings of the macro programs for these functions are included in Appendix B. Instructions for using these functions have been added to the BASIC User's Manual.

In order to run the storage routines the operator must be sure the copy of BASIC on the system disk has been modified to include these functions. When BASIC is downloaded, the message "USER FUNCTIONS LOADED" must appear on the terminal.

The BASIC programs and other files which make up the storage routines are:

DATA.HED	CONFIG.INF	FORMAT.FMT	MACHIN.DAT
PLT200.BAS	OVRNEW.BAS	STO200.BAS	RED200.BAS
CLRLOW.BAS	GRA200.BAS	FIXRED.BAS	DUMMY.BAS

Each file with a .BAS type is a BASIC program which is overlaid onto the main calling program, "OVRNEW.BAS". The result of an overlay is the same as if the new program were typed in from the keyboard. Any old lines with corresponding line numbers in the overlay will be replaced by the new lines.

All variable values and dimension statements are preserved and execution resumes at the next line number following the overlay statement.

The storage programs can be divided into four major functions: the main calling program (OVRNEW.BAS), the plotting program (PLT200.BAS), the the storing program (STO200.BAS), and the data file reading program (RED200.BAS). A listing and discussion of each program is included in Appendix A. Each segment is written so that all overlaying is done between line numbers 20050 and 21350 with increments of 10 in each segment. By doing this, the lower portion of the main program is made common to all segments.

Sec. 4.6 OVRNEW.BAS

The main calling program contains the command address table, most of the simple command subroutines, and an initialization routine which erases itself after execution. This routine, residing in the first portion of the program (lines 10-980), checks that the 11/04 system time and date are set. The routine also checks and updates the "INITIA.LIZ" file on the data disk. If errors are found, they are logged on the terminal and the program halts. The operator must then correct any errors that occurred and re-run the program. When no errors are found, the initialization routine prints a message warning the operator not to use the run command again, without reloading the program from the disk, and prints the hardware configuration file (discussed in Sec. 4.7) into the workspace of the 4025. The initialization routine also sets a default crate address and station number (determined by line 910), and then erases the first 31 lines (10-980) of the program and waits for input from the terminal. A flow chart of this chain of events is shown in Fig. 4-1.

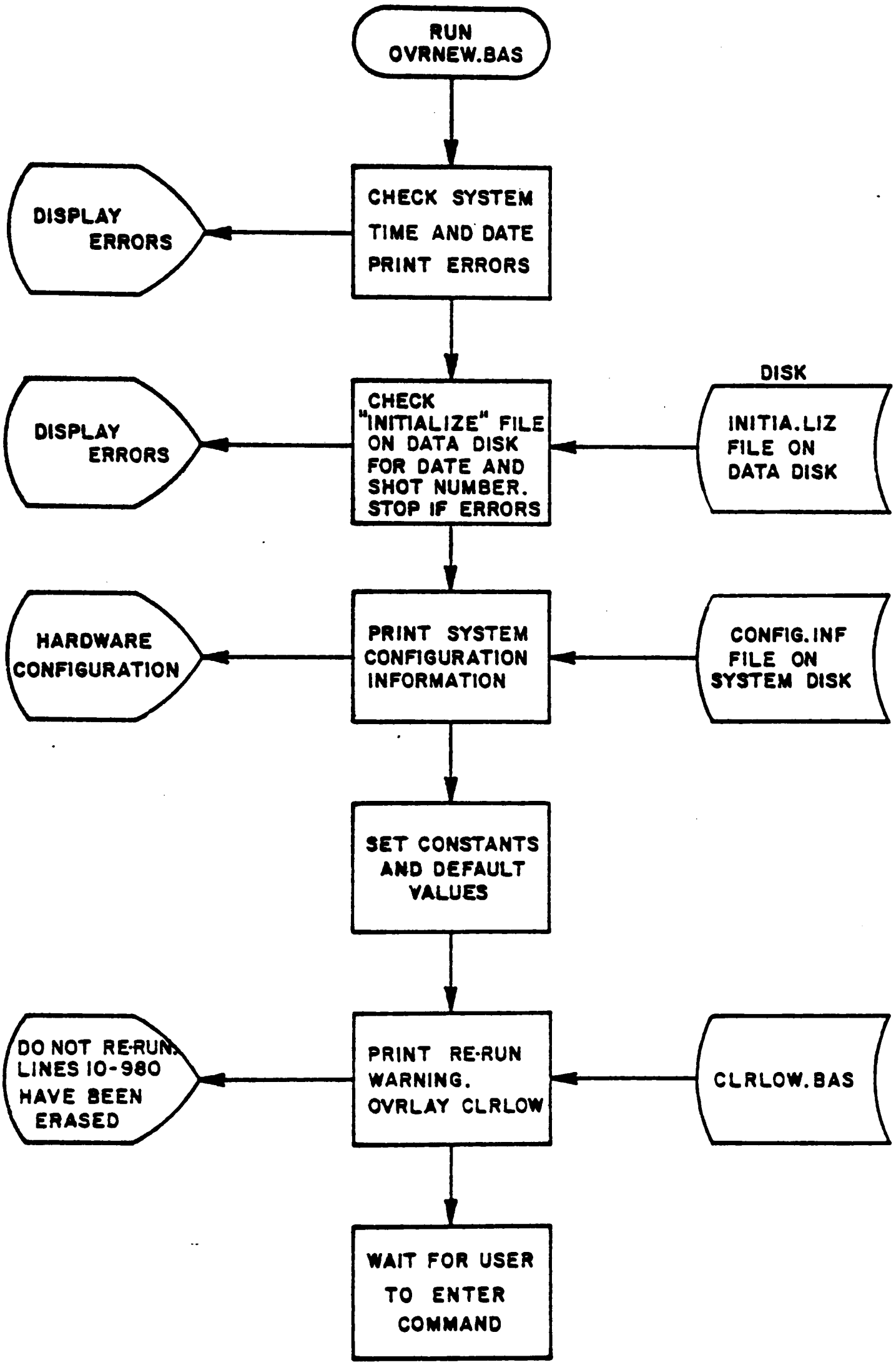

RUN OVRNEW.BAS
CHECK SYSTEM TIME AND DATE PRINT ERRORS
DISPLAY ERRORS
CHECK "INITIALIZE" FILE ON DATA DISK FOR DATE AND SHOT NUMBER. STOP IF ERRORS
DISK
INITIA.LIZ FILE ON DATA DISK
DISPLAY ERRORS
PRINT SYSTEM CONFIGURATION INFORMATION
CONFIG.INF FILE ON SYSTEM DISK
HARDWARE CONFIGURATION
SET CONSTANTS AND DEFAULT VALUES
PRINT RE-RUN WARNING. OVRLAY CLRLOW
CLRLOW.BAS
DO NOT RE-RUN. LINES 10-980 HAVE BEEN ERASED
WAIT FOR USER TO ENTER COMMAND

Fig. 4-1. Flow Diagram of the Initialization Portion of OVRNEW.BAS

Erasing a portion of a program is accomplished by overlaying a file (CLRLOW.BAS) that contains line numbers alone, without statements. As this file is overlayed, each corresponding line in the existing program is erased. Overlays affect only the active memory and cannot modify the file on disk.

Sec. 4.7 Hardware Configuration File

The hardware configuration file "CONFIG.INF" is printed into the workspace for the operator to review and check against the actual configuration of the CAMAC crate. This file contains information about the CAMAC modules and their position within the crate and is used by the store routine. A typical configuration file is shown in Fig. 4-2(a). Each CAMAC module is identified by a unique unit number and by its TTU inventory number. The crate and station numbers describe a module's location within the CAMAC system while the type and bus numbers indicate the kind of equipment used. The post-trigger entry refers to the 2264 digitizers and indicates whether the module is internally jumpered so that the number of post-trigger samples indicated by the front panel switch is read directly, or is multiplied by two (e.g. twice as many post-trigger samples as indicated by the switch). The bus number entry allows the system to include other than CAMAC equipment (583 bus), but no software to support other buses had been developed at the time of writing.

At this point, the program has printed "ENTER COMMAND" on the terminal and is waiting for input. When a command is entered the program tries to match the first three letters (if there are that many) with the three letter commands catalogued in the branch table. If no match is found, the first letter of the entry is compared against single

02-JAN-80 COPY OF CONFIG.INF, OPTION I LPRINT V1:BB

UNIT #	BUS #	CRATE #	SLOT #	MNF. TYP#	POSTRIG	SERIAL #
1,	583,	2,	6,	2264,	1,	169436
2,	583,	2,	16,	2264,	2,	169995
3,	583,	2,	15,	2232.1,	00,	169994
4,	00,	0,	00,	00,	00,	00

(a) Hardware Configuration File

02-JAN-80 COPY OF RAD006.FMT, OPTION I LPRINT V1:BB

FORMT#	#OF REC	DATE								
2001,	6,	791107								
TYPE	UNIT	CHNL	GAIN	OFST	CAT	ID#	1ST-TM	#OF SMPL	LST-TM	STEP
30,	2,	5,	31.1,	1,	0,	0,	-2000,	256,	0,	50
20,	2,	1,	1,	0,	0,	0,	-2000,	256,	0,	50
21,	2,	3,	1,	0,	0,	0,	-2000,	256,	0,	50
113,	1,	2,	1,	0,	105,	4,	0,	1024,	0,	2.5
114,	1,	3,	1,	0,	105,	4,	0,	1024,	0,	2.5
103,	1,	1,	1,	0,	105,	3,	0,	1024,	0,	2.5

(b) Typical Format File

Fig. 4-2. Store Routine Information Files

letter commands in the branch table. If no match is found in either group, the program reprints "ENTER COMMAND" and again waits for input. A list of available commands and their meaning is given in Appendix C.

Sec. 4.8 PLT200.BAS

When OVRNEW.BAS is loaded into memory, it contains the "PLT200.BAS" program residing in lines 20050 to 20990. Note that this is the space where all overlays will occur. The PLT200 program responds to the "G" or "GRAPH" command and will plot data, presently in the CAMAC buffer, on the graphics terminal. The operator has control over the size and scale of the graph, the number of points plotted, and the type of line used (solid, dotted, etc.). Several graphs may be drawn in separate portions of the screen or can be drawn on the same set of axes. A flow chart of the PLT200 program is shown in Fig. 4-3 and a listing and description are included in Appendix A.

Lines 20050 - 20240 of PLT200.BAS form a driving program for a set of general purpose graphics subroutines residing in lines 20250-20990. The graphics routines are written to drive the 4025 graphics terminal. Options, such as log or linear scales, multiple or overlayed graphs, are available. These general purpose routines are normally located in lines 17000-19700 but have been renumbered so as to coincide with the overlay area of the storage routines. Instructions for using these graphics routines have been added to the BASIC Language Reference Manual. The new entry points for the renumbered graphics routines are included in Appendix A.

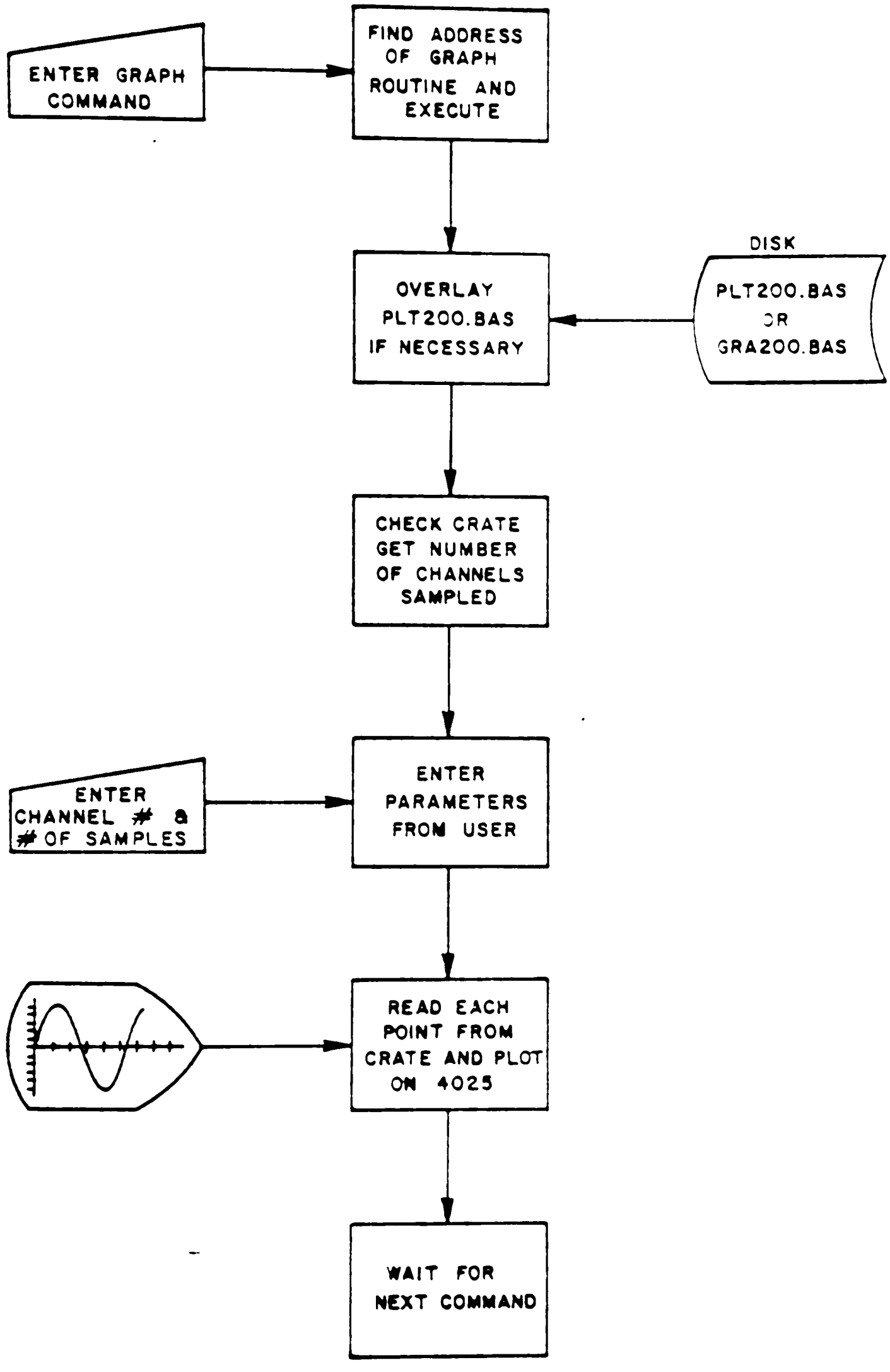

ENTER GRAPH COMMAND
FIND ADDRESS OF GRAPH ROUTINE AND EXECUTE
DISK
PLT200.BAS OR GRA200.BAS
OVERLAY PLT200.BAS IF NECESSARY
CHECK CRATE GET NUMBER OF CHANNELS SAMPLED
ENTER CHANNEL # & # OF SAMPLES
ENTER PARAMETERS FROM USER
READ EACH POINT FROM CRATE AND PLOT ON 4025
WAIT FOR NEXT COMMAND

Fig. 4-3. Flow Diagram of PLT200.BAS

Sec. 4.9 ST0200.BAS and Support Files

The longest and most involved overlay is "ST0200.BAS" occupying lines 20050 to 21350. This routine is downloaded in response to the "STO" or "STORE" command and is capable of reading and storing various types of data from many different sources. The data are stored in virtual files (discussed in Sec. 4.10) on the data disk. A data file can be up to 128 blocks long (about 30,000 data points) provided that sufficient space is available on the disk. The first few lines of each data file contain information describing the length of the file and the I.D. number (version number) of the store program that created it.

For flexibility, the store program acts mainly as an assembler getting most of its information from the CAMAC crate and from four support files located on the system disk. These four files, which may be modified as the experiment changes, include: a hardware configuration file "CONFIG.INF", a machine parameter file "MACHIN.DAT", a collection of descriptive titles for typical Tokamak diagnostics "TITLE.REC", and a file to determine what and how much data is to be stored, called a Format file. Several Format files, corresponding to different experiments, can be stored on the system disk. The operator selects a Format file by entering its title which will always have the form "NAME.FMT" where "NAME" is a 1-6 character file name given by the operator. A flow chart of the store routine is shown in Fig. 4-4.

The configuration file, as discussed in Sec. 4-7, informs the store routine as to what CAMAC hardware is available and its location within the system. Only one "CONFIG.INF" should be on the system disk and should only require modifications when new equipment is acquired or present

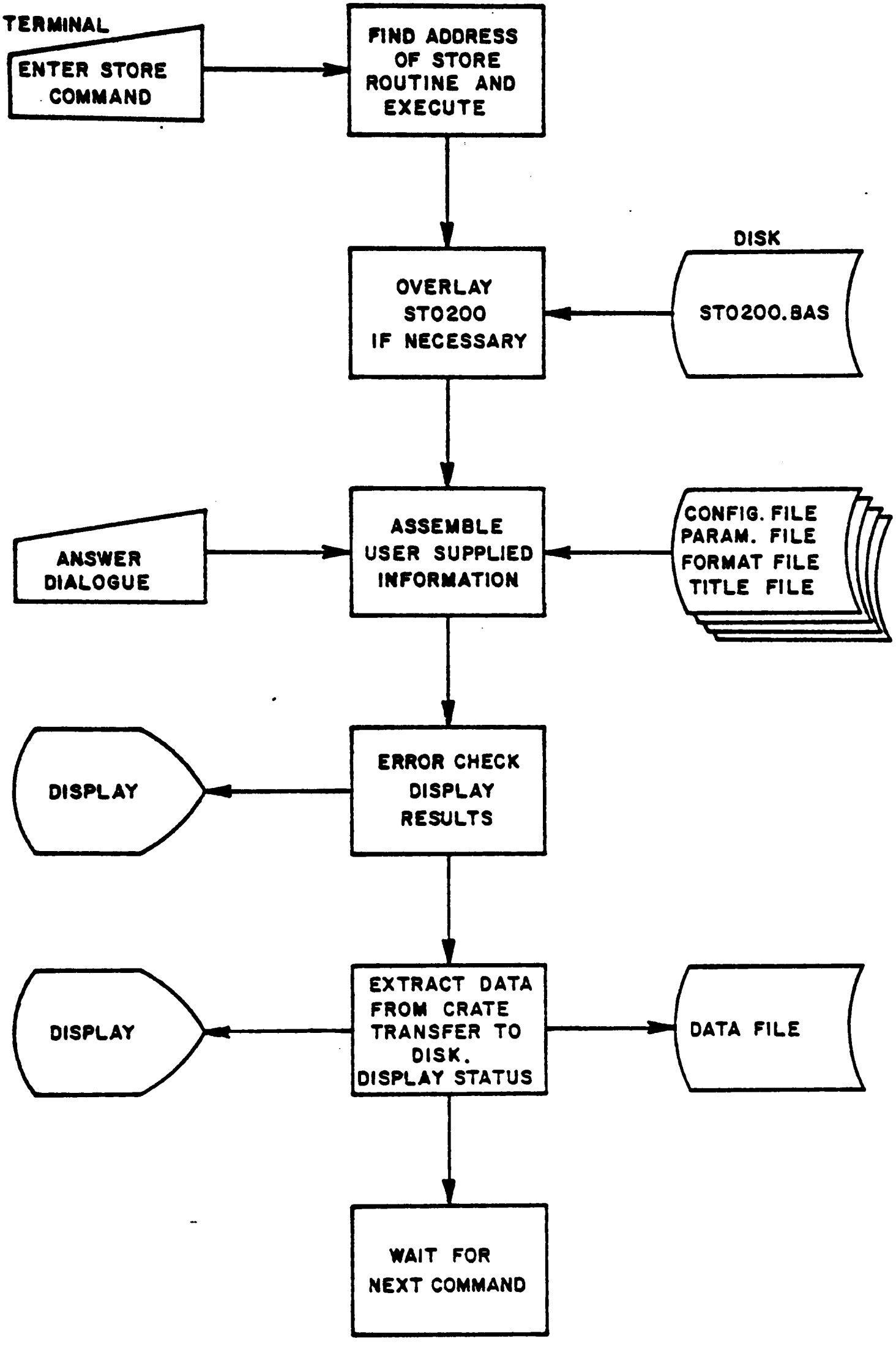


Fig. 4-4. Flow Diagram of ST0200.BAS

equipment is relocated. The configuration file is shown in Fig. 4-2(a).

The record title file "TITLE.REC" contains a three digit "Type" number and title (up to 12 characters) for each diagnostic on the experiment. This file insures that all records of the same type will have exactly the same title in the data file; for example:

030, I PLASMA (plasma current)

020, COS POS (cosine position coil)

If a title for a record is not found in the title file, the message "TITLE NOT FOUND" is written into the data file. Record types and titles may be added to the title file by using the "TITLE" command in the main program. The types and titles presently included are listed in Appendix C.

The parameter file contains information as to how the experiment is set up, such as bank voltages, trigger times, gas pressures, and magnetic field probe data. The parameter file is copied into the workspace of the 4025 terminal, where the operator can modify or replace the file before each shot. A typical parameter file is shown in Fig. 4-5. The version of the parameter file currently in the workspace is used in the data storage procedure.

The first few lines of the parameter file contain information about the size and contents of the file. The abbreviated headings in the second line stand for:

LNS: total number of lines in the parameter file including blank lines and headings

TYP: an I.D. number for this particular parameter file; if modifications are made, this number should be changed

```
02-JAN-80 COPY OF PARM1.LST, OPTION I     LPRINT V1:BB

 1  MACHINE PARAMETERS
 2  LNS TYP CAT LEN OFST
 3  062,1002, 8,  8,  3,
 4
 5  POSITION INFO
 6  CAT OFS TLN DLN
 7  101, 15,  3,  3,
 8  102, 21,  3,  2,
 9  103, 26,  3,  4,
10  104, 33,  3,  5,
11  105, 41,  3,  5,
12  106, 49,  3,  1,
13  107, 53,  3,  1,
14  900, 56,  2,  2,
15  =======LAST LINE OF PARAMETER FILE HEADER=====START USER INFORMATION======
16
17  CHAMBER
18    GAS ID #  PRESSURE(T)  GAS NAME
19           999,   2.00E-07,BASE
20           101,   5.80E-03,D2 FILL
21            00,   0.00E-00,
22
23  PUFF FILL
24  VALVE ID #  TIME (+T0)  PRESSURE(T)  GAS ID #    GAS NAME
25            01,          00,    0.0E-00,         101,    D2
26            00,          00,    0.0E-00,          00,
27
28  CAMAC TRIGGER TIMES (USEC)
29    UNIT #   TRIG.TIME(+T0)
30             1,       13300
31             2,       13300
32             3,          00
33             4,          00
34
35  BANK VOLTAGES (KV)
36    BANK ID #,    FAST    ,    SLOW    TRIG(+T0)USEC BANK NAME
37           101,        1.1,        0.5,       13300,OHMIC HEAT.
38           102,       0.20,       0.06,       13300,VERT. FIELD
39           103,       13.0,          0,       11600,PREIONIZER
40           104,        6.6,          0,        3150,TOR. FIELD
41           105,    4.0E-06,          0,        3150,RAD.(HENRY)
42
43  PROBES
44  PROBE ID#  ,TOR. POS(D),POL. POS(D),RAD.POS(CM),ORIENTATION
45             3,          60,         180,          13,           0
46             4,          60,         270,           0,           0
47              ,           ,           ,            ,
48              ,           ,           ,            ,
49              ,           ,           ,            ,
50
51  ANTENNA
52  ANTEN. ID# ,TOR. POS(D),POL. POS(D), FREQ.(MHZ),CURRENT(A PP)
53           101,          0.0,         180,          8.0,         80.0
54
55  SPECTROMETER
56  SPEC. ID#    WAVELEN.(A),PM VOLT(KV),SLIT 1 (UM),SLIT 2 (UM), FILTER ID#
57           101,     5461.0,          0.0,         10.0,         10.0,          100
58
59  COMMENTS
60  TYPICAL FAST WAVE PROFILE SHOT
61
62  END PARM FILE
```

Fig. 4-5. Typical Parameter File

CAT: the number of categories in the user information section

LEN: the length (number of data lines) of the position information category

OFST: the offset to the first data line of the position information category

The abbreviated headings in the position information category (line 6) are:

CAT: the category number of each section of the user information; the comment section (900) should always come at the end of the file

OFST: the offset to the first data line of that category

TLN: number of lines in the category title

DLN: number of data lines in the category

The offset numbers in these two categories are used by data retrieval and analysis routines to access data quickly. The offset numbers are calculated from the equation:

OFST = (line number of first data line in that category)
-(line number of first data line in parameter file) - 1

In the example in Fig. 4-5, the first offset number is calculated as:

$$OFST = 7 - 3 - 1 = 3 .$$

The offset numbers are used to calculate the I_{th} data line of a category from the equation:

line number of I_{th} line = (line number of first data line in parameter file) + OFST + I

As an example of this procedure, the line number of the second data line in the fourth category (BANK VOLTAGES) of the file in Fig. 4-5 is:

$$\text{line number} = 3 + 33 + 2 = 38$$

In the data file all lines are numbered, including blank lines and heading lines so that the line number calculated above is the 38th line in the file. Through these offset numbers, the position of each category, relative to the first data line, can be determined. The line number of the first data line in the parameter section is recorded in the first data line of each data file for easy access.

The rest of the parameter file gives information about the experimental setup. All trigger times are relative to t_0 which is the initial trigger pulse for the experiment. Where more than one similar piece of equipment exists, as with magnetic probes, each is given a 1 to 3 digit I.D. number. The position of a probe around the Tokamak is determined by a toroidal and poloidal angle, measured in degrees, and the distance from the tip of the probe to the center of the chamber as shown in Fig. 4-6. The orientation of the probe refers to the angles that the Normal vector of the probe coil makes with the horizontal plane (elevation) and the toroidal, or "Z", direction (azimuth). A single number is used to represent both angles and is calculated by the equation:

$$\text{number} = 360 \times AZ + EL$$

where "AZ" is the azimuth angle, and "EL" is the elevation angle as shown in Fig. 4-6 (both angles are integers expressed in degrees).

It is important to remember that the information in the parameter file is entered by the operator and not measured by the system. As the data acquisition system expands, more of this information will be obtained by the computer, but because there are so many important parameters to be recorded, it is difficult to envisage a significant

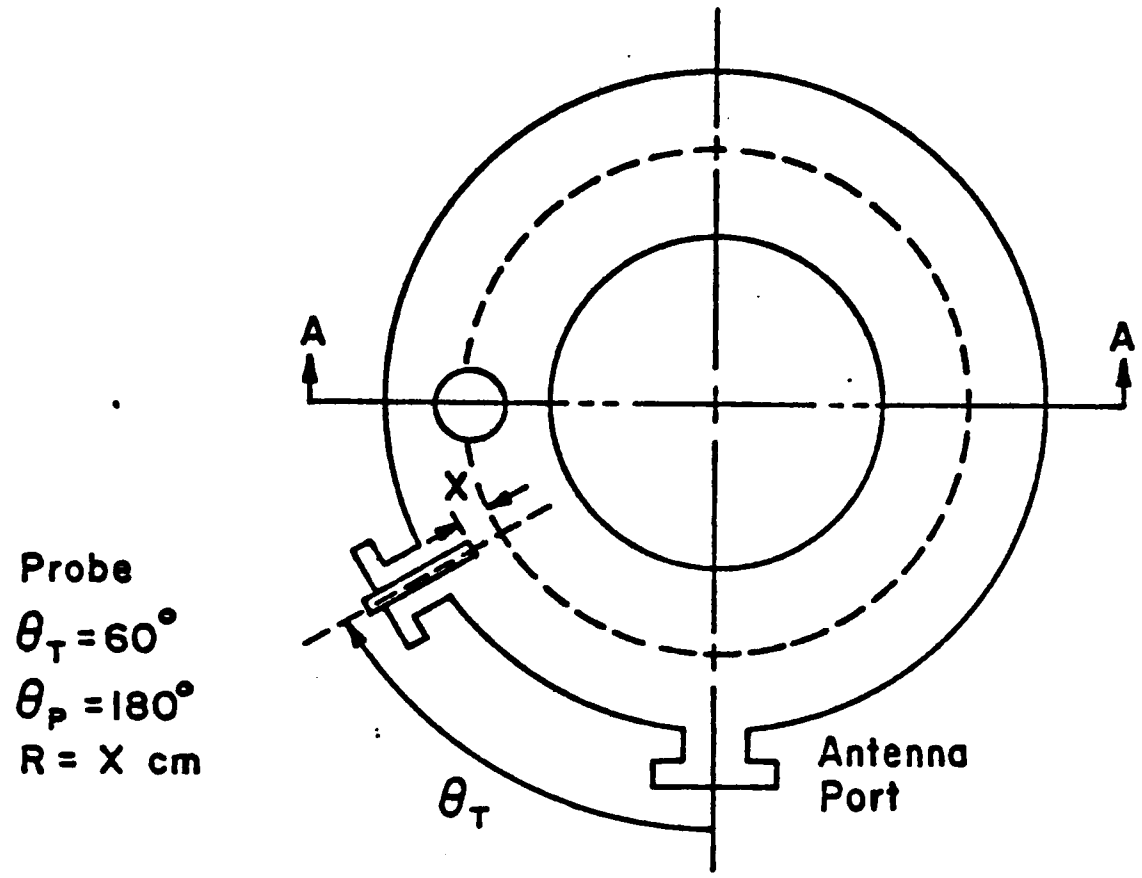


Fig. 4-6. Probe Position and Orientation

decrease in the size of the parameter file. The parameter file is added to each data file for use by reading and analysis programs.

The last file accessed by the store routine is the Format file. A typical Format file is shown in Fig. 4-2(b). This file is created by the operator to determine which channels and how many data points per channel are being stored in the data file. The first data line of the Format file contains a 4 digit I.D. number to identify this particular Format file. Also on this line is the number of records being stored and the creation date of the Format file. The rest of the file contains a single line for each record being stored and includes:

TYPE: a 3 digit number associated with each diagnostic measurement being made on the Tokamak; record types and titles are stored in the "TITLE.REC" file on the system disk

UNIT: the unit number of the module that digitized the data

CHNL: the channel number (if applicable)

GAIN: the overall gain (+/-) of any circuitry between the transducer and the digitizer (i.e. attenuators, amplifiers, etc.)

OFST: any voltage offset induced between signal source and digitizers

CAT: category refers to a section of the parameter file where more specialized information will be found (if applicable)

ID#: used to differentiate between similar pieces of equipment, such as various probes

1ST-TM: the time of the first data point (in μs relative to the module trigger time)

#OF SMPL: the number of data points to store

LST-TM: time of the last data point (in μs relative to the module trigger time

STEP: time between stored data points (in μs)

The last two entries are mutually dependent, the operator enters the desired value for one and zero for the other. The program computes and enters the missing value into the data file. The commas in these files and the data file header allow the information to be read by BASIC "INPUT" statements and "free format" FORTRAN "READ" statements. Consistent right justified alignment allows for reading by formatted FORTRAN "READ" statements.

After assembling the information from these four files, the store program will check to see if the values are compatible with the present hardware configuration. If any discrepancies are found, they are printed on the screen and the operator is asked to correct them. When the assembly is completed correctly, the store program will extract the data from the CAMAC buffers and deposit the data, along with the parameter and format information, into a data file on disk DX1. When finished, the title and length of the data file are printed on the terminal and the program returns to the "ENTER COMMAND" loop in the main program.

Sec. 4.10 The Data File

A data file, as created by the storage program, consists of 3 parts. The first part contains directory information about the size, contents, and organization of the data file. For a single sample record, this first section would contain the data word as well. The second part is a copy of the parameter file. The final part of the data file contains

the binary data for each record stored in virtual arrays. These arrays allow the program to use the disk as a block of random access memory. Data are passed to and from the array in the form of subscripted variables. By this method, large amounts of data (that would normally overflow the active memory) are made available to the program. The combination of the first two parts make up the data file "header". The header of each data file is in the form of an ASCII text file and can be listed on the terminal. A typical data file header is shown in Fig. 4-7. This particular data file contains 6 records, stores 3840 data points, and requires 23 blocks of disk space.

Each line of the data file header begins with a line number to facilitate reading and analysis programs. In the example header of Fig. 4-7, lines 1001 to 1017 make up the directory section of the file. The information in lines 1001 and 1002 includes:

F#: a three digit number identifying the store routine that created the data file

FFL#: the I.D. number of the Format file used

SHT#: the shot number (starts at zero with each new date)

#REC: the number of records in this file

CNT: continuation code; allows data files to be chained together if necessary (e.g. 12 means the first of two files).

CPY: used to distinguish between different copies; the master copy is "1"; copies that are different (corrections have been made) are numbered 2,3 etc.

DATE: the date the data file was made in a YYMMDD format as defined in Sec. 4-4

```
02-JAN-80 COPY OF TYPICA.HED, OPTION I    LPRINT V1:BB

1001, F# FFL#  SHT#  #REC  CNT  CPY    DATE   REC.LST  PARM.LST   TIME   #LINES
1002,101,2001,    1,    6,  00,   1,  800102,   1005,    1020,    59856,   79,
1003,TIME 16:37, STEVE, TYPICAL DATA FILE
1004,TYPE  CAT  ID#  HEADR   LEN     TITLE
1005,  30,   0,   0,  1012,     1,
1006,  20,   0,   0,  1013,     1,
1007,  21,   0,   0,  1014,     1,
1008, 113, 105,   4,  1015,     1,
1009, 114, 105,   4,  1016,     1,
1010, 103, 105,   3,  1017,     1,
1011, BLK#  1ST-TM   STEP #SMPL  LST-TM   GAIN OFSET ERFLG CKSUM SRIAL#
1012,    9,  -2000,    50,  256,  10750,   31.1,    1,   0,100000,169995,
1013,   10,  -2000,    50,  256,  10750,      1,    0,   0,100000,169995,
1014,   11,  -2000,    50,  256,  10750,      1,    0,   0,100000,169995,
1015,   12,      0,   2.5, 1024, 2557.5,      1,    0,   0,100000,169436,
1016,   16,      0,   2.5, 1024, 2557.5,      1,    0,   0,100000,169436,
1017,   20,      0,   2.5, 1024, 2557.5,      1,    0,   0,100000,169436,
1018,MACHINE PARAMETERS
1019,LNS TYP CAT LEN OFST
1020,062,1002, 8,  9,  3,
1021,
1022,POSITION INFO
1023,CAT OFS TLN DLN
1024,101, 15,  3,  3,
1025,102, 21,  3,  2,
1026,103, 26,  3,  4,
1027,104, 33,  3,  5,
1028,105, 41,  3,  5,
1029,106, 49,  3,  1,
1030,107, 53,  3,  1,
1031,900, 56,  2,  2,
1032,=======LAST LINE OF PARAMETER FILE HEADER=====START USER INFORMATION======
1033,
1034,CHAMBER
1035,  GAS ID #  PRESSURE(T)  GAS NAME
1036,         999,   2.00E-07,BASE
1037,         101,   5.80E-03,D2 FILL
1038,          00,   0.00E-00,
1039,
1040,PUFF FILL
1041,VALVE ID #  TIME (+TO)  PRESSURE(T)  GAS ID #    GAS NAME
1042,          01,          00,    0.0E-00,         101,    D2
1043,          00,          00,    0.0E-00,          00,
1044,
1045,CAMAC TRIGGER TIMES (USEC)
1046,  UNIT #   TRIG.TIME(+TO)
1047,           1,       13300
1048,           2,       13300
1049,           3,          00
1050,           4,          00
1051,
1052,BANK VOLTAGES (KV)
1053,  BANK ID #,     FAST    ,     SLOW    TRIG(+TO)USEC BANK NAME
1054,         101,         1.1,         0.5,       13300,OHMIC HEAT.
1055,         102,        0.20,        0.06,       13300,VERT. FIELD
1056,         103,        13.0,           0,       11600,PREIONIZER
1057,         104,         6.6,           0,        3150,TOR. FIELD
1058,         105,    4.0E-06,           0,        3150,RAD.(HENRY)
1059,
1060,PROBES
1061,PROBE ID#  ,TOR. POS(D),POL. POS(D),RAD.POS(CM),ORIENTATION
1062,           3,          60,         180,          13,           0
1063,           4,          60,         270,           0,           0
1064,           ,           ,           ,            ,
1065,           ,           ,           ,            ,
1066,           ,           ,           ,            ,
1067,
1068,ANTENNA
1069,ANTEN. ID# ,TOR. POS(D),POL. POS(D), FREQ.(MHZ),CURRENT(A PP)
1070,         101,         0.0,         180,          8.0,         80.0
1071,
1072,SPECTROMETER
1073,SPEC. ID#   WAVELEN.(A),PM VOLT(KV),SLIT 1 (UM),SLIT 2 (UM), FILTER ID#
1074,         101,     5461.0,         0.0,         10.0,         10.0,         100
1075,
1076,COMMENTS
1077,TYPICAL FAST WAVE PROFILE SHOT
1078,
1079,END PARM FILE
```

Fig. 4-7. Typical Data Header

REC.LST: line number of the first data line in the record list section (lines 1004 to 1010 in example)

PARM.LST: line number of the first data line in the parameter section (to be used with the offset numbers)

TIME: time that data was stored (seconds after midnight)

#LINES: the total number of lines in the data file header.

Line 1003 gives the time again in "HH:MM" form, the operator's name, and a comment (up to 57 characters) if the operator wishes to enter one. The record list section (lines 1004 to 1010) gives a brief summary of the records in the file and the addresses of where to get more information on each record. The "TYPE", "ID#" and "TITLE" information is the same as before. The "CAT" entry is the category of the parameter section pertaining to that record (if applicable). The "HEADR" and "LEN" entries give the start address and length, respectively, of more complete information about each record. This information is called the record header (lines 1011 to 1017). Each entry is described as follows:

BLK#: the starting block number, within the data file, of the binary data array for that record

The "1ST-TM", "STEP", "#SMPL", "LST-TM", and "GAIN" entries are the same as for the Format file information. The "SERIAL#" entry is the TTU inventory number of the module that took the data. The "ERFLG" and "CKSUM" entries are not presently used and are loaded with dummy values. These will be used in future versions with the following meaning:

ERFLG: indicates if any of the data in that record are erroneous; typical errors are (ERFLG values in decimal):

ERFLG = -1: the error checking logic was not implemented

= 0: no errors detected

= 1: some fullscale readings are present; these may represent overloads (e.g. all 0's or all 1's)

= 10: some samples have been replaced by dummy values (outside the valid range) because data were not available at the time.

This indication is somewhat redundant, because the errors covered can be detected by just checking the data. However, it is very useful in FORTRAN and BASIC programs, because error checking could double or triple the time required for data retrieval. "ERFLG = 0" allows the reading program to bypass error checks for all of that record.

CKSUM: a "checksum" used to verify that data has been read correctly-from the disk. The "CKSUM" value is defined as:

$CKSUM = 2^{16}$ - (sum of all data words in that record) calculated modulo 2^{16}

The data words are interpreted in the same way as integer virtual array elements in BASIC, and all words including errors are summed. The present value (100,00) is an impossible value and indicates that checking was not performed by the storage routine. This checksum is not connected with the checksum employed (transparently) by the RT-11 operating system.

If a record contains single sample data, the data word is stored in the header section and the columns of line 1011 of Fig. 4-7 take on new meaning.

BLK#: is set to zero indicating that no virtual data array exists for this record.

1ST-TM: is the time (in μs relative to the module trigger time) that the sample was taken

STEP: contains the data word

The "#SMPL" and "LST-TM" columns are defined the same way as "1ST-TM" and and "STEP", respectively, so that a single record might contain two single data samples taken at different times. For single sample data, line 1011 of Fig. 4-7 would read:

"1011, BLK# 1ST-TM DATA1 2ND-TM DATA2 ..."

The rest of the columns are the same as before.

The length of the data file header depends upon the number of records in the file. The number of records is limited by the dimensions of several arrays, and this is limited by the amount of active memory in the system. With 32 k-bytes of memory, a maximum of 10 records can be stored in each shot. If more memory were added, this limit could be increased (by about 30 records/k-byte added) by changing the dimension statements in lines 920 to 930 of "OVRNEW.BAS". (Note that the dimension of "L6" is one greater than that of the other arrays on these lines.)

The maximum number of data points that can be stored depends upon the amount of available disk space (up to a maximum of about 30,000 points) where each block of disk space can store 256 points. A newly formatted data disk has 473 free blocks.

Sec. 4.11 Temporary Files

Two other files are created on the system disk by the ST0200 program. The "DATA.HED" file is a temporary sequential[8] ASCII file containing the data file header information until the data file is created.

The second file arises from the problem that the BASIC "OPEN" statement used to create disk files will not accept a variable for its size specification and the size of the data file will vary from shot to shot. To get around this problem, once the size of the data file has been calculated, the basic statement that will open the data file is stored, along with a line number and the proper data file size, as a one line overlay, in a file called "DUMMY.DAT". The dummy file is then overlayed onto "STO200" which writes the "OPEN" statement into a later part of the program. The "DATA.HED" and "DUMMY.DAT" files are rewritten each time a data file is stored.

Sec. 4.12 RED200.BAS

The final overlay in the storage routines is "RED200.BAS" which is used to plot data from data files onto the 4025 terminal. Called by the "READ" command, "RED200" executes in much the same way as "PLT200" except that the data come from a disk file rather than from the CAMAC crate. Also, with the "READ" command, the directory portion of the data header is printed out to the monitor space of the 4025 terminal. A flow chart of "RED200" is shown in Fig. 4-8 and a listing of the program is included in Appendix A.

Lines 20050 to 20990 of "PLT200" and "RED200" are the same. If one of these overlays is present, only the top 20 lines need to be overlayed to change to the other function, resulting in a faster response. For this purpose, two more overlays are included in the system, "GRA200.BAS" and "FIXRED.BAS" which contain the top 20 lines of "PLT200.BAS" and "RED200.BAS", respectively.

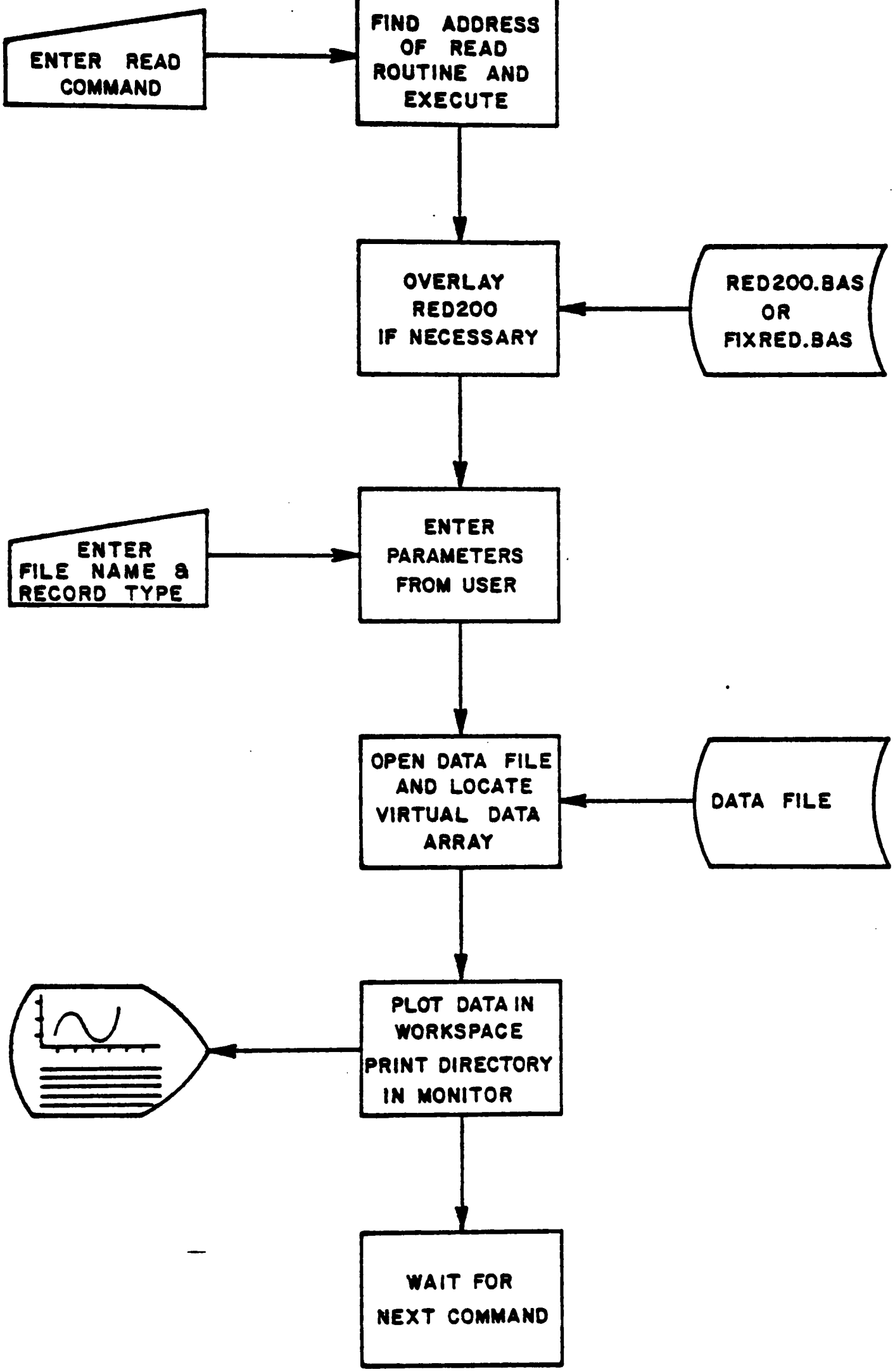

ENTER READ COMMAND
FIND ADDRESS OF READ ROUTINE AND EXECUTE
OVERLAY RED200 IF NECESSARY
RED200.BAS OR FIXRED.BAS
ENTER FILE NAME & RECORD TYPE
ENTER PARAMETERS FROM USER
OPEN DATA FILE AND LOCATE VIRTUAL DATA ARRAY
DATA FILE
PLOT DATA IN WORKSPACE
PRINT DIRECTORY IN MONITOR
WAIT FOR NEXT COMMAND

Fig. 4-8. Flow Diagram of RED200.BAS

CHAPTER V

OPERATING INSTRUCTIONS

Sec. 5.1 Introduction

This chapter is intended to be used as a set of instructions for bootstrapping[9] the computer and using the storage routines on the 11/04 system.

Sec. 5.2 Turn-on and Bootstrap Procedure

Turn on the AC power to all equipment. The order is not important, but the 4025 terminal requires about 15 seconds to warm up before it can be used (e.g. no cursor appears). The equipment requiring AC power is:

a. the computer (front panel)

b. the terminals (if used)

c. the CAMAC crate (power supply must be turned on to use the computer for any purpose)

d. optical isolator (two separate supplies)

e. the communication interface (if installed)

f. Tektronix 604 monitor (if used)

At this point the computer responds only to capital letters. Both terminals have keys to "lock on" capitals without having to press the "Shift" key. On the LA36 depress the "Cap's Lock" key (key should stay down) and on the 4025 terminal depress the "TTY Lock" key (indicator light should be on). Set terminal select switch to "Graphics" and adjust baud rate if necessary (see 4025 User's Manual and Sec. 3.3). Set the communications interface for "TTY↔PDP" (see Fig. 3-14) with:

Monitor all data: disabled

TTY2: disabled

TTY1: enabled

Place the storage routine system disk in drive DX0. At the front of the computer, place the "Local/Remote" switch in the "Local" position and the"Halt/Continue" switch in the "Continue" position. Press and release the "Boot" switch. The computer should print four numbers on the terminal:

"$xxxxxx xxxxxx xxxxxx xxxxxx" ("_" is the cursor on the 4025 or the "$_" print head position on the LA36)

If the system were being rebooted after a crash, these four numbers would be the contents of memory registers that would indicate what the computer was doing when it crashed. To run the boot routine, type in:

DX (CR) ("CR" is carriage return)

For this step, the computer output should be directed to the 4025 (or its "Echo" turned on) because the RT-11 initialization command file "STARTS.COM", which is executed by the bootstrap program, will "Learn"[10] several special purpose keys on the 4025 keyboard. When the boot routine has finished (≈10 s) the computer responds with a period "." on the terminal. This is the RT-11 "Prompt" character signifying that the system is booted and the RT-11 monitor is resident and awaiting input.

Sec. 5.3 RT-11 Commands

RT-11 commands are discussed in the RT-11 User's Manual. Use the "DATE" and "TIME" commands to set the system clock and calendar. Type in:

DATE DD-MMM-YY (CR) e.g. DATE 15-DEC-79

TIME HH:MM:SS (CR) e.g. TIME 13:29:00 (24 hour clock)

Any other RT-11 commands can be executed at this time.

Sec. 5.4 Creating a Data Disk

A specially formatted data disk must be used with the storage routines. The disk must first be "Initialized" as described in the RT-11 System Generation Manual. While in RT-11, place a blank disk in drive DX1 and use the RT-11 "INITIALIZE" command to create a directory file on the disk. Type in:

INITIALIZE DX1: (CR) .

Since this will erase any data previously on the disk, the system asks:

"INITIALIZE DX1: ARE YOU SURE?" .

Answer:

YES (CR) .

All new disks must be initialized as described above before they can be used on the 11/04 system. After initializing each disk, type:

DIRECTORY/BAD DX1: (CR)

to test the disk for bad blocks. The special data disk format is created by a BASIC program called "CREATE.BAS" residing on the system disk. Before a BASIC program can be run, the BASIC language interpreter must be downloaded into active memory. Type:

R O (CR) . (O.SAV is the name of the BASIC language on this system disk. Typing the ".SAV " is optional)

The system responds with:

"USER FUNCTIONS LOADED"

"*_" .

Type a carriage return to include all basic functions (see BASIC manual for description of functions). The system responds with:

"READY"

"_"

which is the BASIC "prompt" sequence indicating that BASIC is resident and ready for input. At this time any BASIC command can be executed (BASIC commands are discussed in the BASIC User's Manual).

When the boot routine is executed, the RT-11 "ASSIGN" command is used to make DX1 the default disk, that is, any command without a specific address is directed to DX1. To load the "CREATE.BAS" program from the system disk (DX0), type:

OLD 'DX0:CREATE' (CR) (the ".BAS" is optional) .

When the system responds with "READY", Type:

RUN (CR)

to execute the program. The "CREATE.BAS" program, as discussed in Sec. 4.4, gives detailed instructions to the user. Several disks can be formatted at this time but they must have been previously initialized by the RT-11 monitor. A listing and functional description of "CREATE.BAS" is included in Appendix A.

Sec. 5.5 Running the Storage Routines

The BASIC language should be resident as described in Sec. 5.4. Use the BASIC "OLD" command to load "OVRNEW.BAS" from the system disk. Type:

OLD 'DX0:OVRNEW' (CR) .

The program is long and will take approximately 50 s to load. The system will type "READY" when finished. Type:

RUN (CR)

to execute the program. The first portion of "OVRNEW" will check that the time and date have been set and that a data disk is in drive DX1. If these conditions are not met, the operator is asked to return to RT-11 (by typing control C) and make corrections.

When no errors are found the "OVRNEW" program prints a message about reentering the program, prints the configuration file on the terminal, and asks for the operator's name and the name of the parameter file. At this point the data storage routines have been initialized. The message "ENTER COMMAND" is printed and the program is now ready to execute any of the commands listed in Sec. 5.7 and Appendix C.

As mentioned earlier, the first portion of the program has been erased to conserve memory space. Because of this, the BASIC "RUN" command cannot be used to restart the program after a halt or crash. If the program halts, but the program and the BASIC language are still resident, the program can be restarted by typing "GO TO 1000" (CR). The BASIC "GO TO" command leaves all variables unchanged and any open disk files are left open, so it is usually wise to type "CLOSE" (CR), then "GO TO 1000" (CR), if problems are encountered during store or read operations.

Sec. 5.6 CAMAC "F" Commands

Before issuing a CAMAC command to a module, the proper crate and station numbers must be set by using the system "CRATE" command, as described in Sec. 5.7 and Appendix C. The crate number must correspond to the thumbwheel switch on the crate controller (always set to 2 in the present system). The station number (1 to 25) determines which module is being addressed. The present setting for crate and station numbers can

be examined with the "ENQUIRE" command. Once the crate and station numbers are set, CAMAC commands are issued with the "F(n)" command (see Sec. 5.7 and Appendix C). The response to an "F(n)" command will depend upon the particular module and its function. For example; with the 2264 8 channel digitizer an F(9) will start the digitizer, and an F(26) will stop the digitizer (this can also be done from the front panel of the 2264). CAMAC commands and their responses are discussed in the User's Manual for each module.

If the system command being executed requires a new program segment to be overlaid, the message "WAIT (PROG)", where "PROG" is the name of the segment, is printed on the terminal. A 5 to 15 s delay (depending on the length of the segment) is incurred before execution as the segment is loaded into memory. Once loaded, a segment remains in memory until another segment is required. The commands which call for overlays are: "GRAPH" (PLT200), "STORE" (STO200), and "READ" (RED200). All other system command routines are contained in the main program and are available to all overlays.

Sec. 5.7 System Commands

There are presently 26 system commands which can be executed in response to the "ENTER COMMAND" prompt from the main calling program. Commands are recognized either by the first three letters or the first letter entered (the rest of the word is optional, e.g. "STOP" would be interpreted as "STORE"). The entered command is checked first against the three letter entries in the address table (lines 1010 to 1100 in "OVRNEW.BAS"). If no match is found, the command is checked against the single letter commands. The available system commands are listed

below and in Appendix C.

Commands that include numbers such as the F# or V# commands must be typed exactly as illustrated (e.g. the entire word cannot be typed. In the following examples, (CR) indicates the carriage return key and any questions or responses generated by the computer are enclosed in quotes (e.g. "RESPONSE").

Three Letter Commands

Only the first three letters of these commands need to be entered for the command to be recognized by the program.

AUT#: Automatically issues start and stop triggers, separated by a delay, to the 2264 digitizer which allows a single trace on the 604 monitor for each sample period. This is useful for determining the settings for the front panel switches to get the desired data window and sampling rate. The delay is presently about 30 ms. If a very slow sampling rate or large number of post trigger samples is desired, it may be necessary to add a second delay between the start and stop trigger codes (F9 and F25 in lines 1400 and 1410) to allow the 2264 to fill its buffer.

ERASE: clears the multiple and overlay flags in the graphics routines so that the next graph will erase the workspace (no immediate action)

example: ERA (CR)

NAME: enters operator's name

example: NAM (CR)

"ENTER OPERATOR'S NAME"

JOHN (CR)

PARAMETER: prints a parameter file into the workspace of the 4025 terminal and allows the operator to make changes, in "form fillout"[11] mode, and save the new file (or replace the old file by using the same name). The parameter file presently in the workspace at the time the STORE command is issued will be added to the data file.

example: PAR (CR)

"PARAMETER FILE?"

ANY NAME (CR) use W if file is already in the workspace

"MAKE CHANGES NOW" (by moving cursor to work-space, return cursor to monitor when finished)

"SAVE NEW FILE?"

YES (CR)

"NEW NAME?"

ANY NAME (CR) using the same name will replace the old file

READ: downloads "RED200.BAS" (if necessary) and plots data records from files on the data disk onto the 4025 terminal; Any available graphics options (OVERLAY, WIDTH, etc.) may be used but must be set before the READ command is issued.

example: REA (CR)

"WAIT (READ)"

"FILE NAME?"

791206.003 (CR) shot 3 on 12/6/79

"TYPE?"

030 (CR) type 030 is plasma current

The data are plotted in the workspace and the data file header is typed in the monitor. This command does not offer the "first sample" or "number of samples" options used in the GRAPH command.

SCALE: changes the vertical scale factor for the next graph (no immediate action); default values are ±130 and are determined by line 910 in OVRNEW.BAS

example: SCA (CR)

"ENTER YMIN, YMAX"

-50, 100 (CR)

SHOT: allows the operator to modify the shot counter

example: SHO (CR)

"SHT# = n" n = the last shot stored

"ENTER SHOT#?"

ANY NUMBER (CR)

STATUS: prints out the DCC-11 status word (see the DCC-11 manual)

example: STA (CR)

"LAM Q X F_3 F_2 F_1" CAMAC status information printed both as shown and as a binary word

STORE: overlays and runs the storage routine

example: STO (CR)

"WAIT (STORE)"	if STO200.BAS is not already present
"FORMAT FILE?"	
W, or WS, or ANY NAME (CR)	W will use the Format file presently in the workspace; WS will save (under a new name if desired) and then use the file presently in the workspace; ANY NAME will load a Format file into the workspace from the system disk.
"MAKE CHANGES NOW"	only if a new file is being loaded from disk
"SAVE NEW FORMAT?"	
YES (CR)	
"FILE NAME?"	
ANY NAME (CR)	using the same file name will replace the old file
"COMMENT?"	
ANY COMMENT (CR)	up to 57 characters

After this point, no more input is required from the operator but the program will print several status messages as it executes.

STORE: overlays and runs the storage routine

example: STO (CR)

"WAIT (STORE)"	if STO200.BAS is not already present
"FORMAT FILE?"	
W, or WS, or ANY NAME (CR)	W will use the Format file presently in the workspace; WS will save (under a new name if desired) and then use the file presently in the workspace; ANY NAME will load a Format file into the workspace from the system disk.
"MAKE CHANGES NOW"	only if a new file is being loaded from disk
"SAVE NEW FORMAT?"	
YES (CR)	
"FILE NAME?"	
ANY NAME (CR)	using the same file name will replace the old file
"COMMENT?"	
ANY COMMENT (CR)	up to 57 characters

After this point, no more input is required from the operator but the program will print several status messages as it executes.

SUBADDRESS: allows the operator to modify the CAMAC subaddress

example: SUB (CR)

"ENTER SUBADDRESS?"

n (CR) n = 0 - 15

TITLE: adds or changes record titles in the "TITLE.REC" file

example: TIT (CR)

"TYPE"

nnn (CR) nnn = record type number

"TITLE"

ANY TITLE (CR) up to 12 characters

Single Letter Commands

Only the first letter of these commands needs to be entered.

A#: advances through the 2264 buffer memory by executing a series of (#) F(2) commands, the $(\# + 1)_{th}$ data word can then be read with the R command.

example: A# (CR) # = 1 - 32767 data words

CRATE: allows user to modify the default crate and station numbers used by the system commands (the STORE, PLOT, and READ routines set these values automatically)

example: C (CR)

"ENTER CRATE, STATION#?"

2, 16 (CR) address module in station 16 of crate 2

DATA: changes the number of data points used in the graphics routines; the 4025 graphics terminal creates plots by connecting a fixed number of data points with straight

lines. The default number of points is 100. Using more points will increase the resolution of the plot but will require more time to draw.

example: D (CR)

"ENTER # DATA POINTS"

200 (CR) all subsequent graphs will use 200 points

ENQUIRE: returns the present crate, station, and subaddress numbers (set by the CRATE and SUBADDRESS commands)

example: E (CR)

"CRATE 2 STATION 6 SUBADDRESS 0"

F#: executes a CAMAC F code; the crate, station, and subaddress must be properly set prior to using this command.

example: F9 (CR) # = 0 - 31

"F9"

"THE RESULT" if applicable

GRAPH" overlays and runs PLT200.BAS to plot data presently stored in the buffer memory of the 2264 digitizers (proper station number must be set).

example: G (CR)

"CHANNEL?"

n (CR) n = 1 - 8

"# OF SMPLS, 1ST SMPL?"

n, s (CR) n, s = 0 - 17000

SMPLS in this command refers to the data in the buffer. The plot can be started at any point in the buffer. This

makes it possible to expand any portion of a plot.

HALT: stops execution of the program and returns to the BASIC language: the program can be restarted with the "GO TO 1000" command

example: H (CR)

"READY" (BASIC prompt)

INITIALIZE: executes a CAMAC Z command (see Sec. 2.6), starts all digitizers (any previous data are lost)

example: I (CR)

"INITIALIZE Y/N?"

YES (CR)

MULTIPLE: sets the multiple flag in the graphic section; all subsequent graphs will be placed in different areas of the screen without erasing previous data

example: M (CR) (no response from computer)

NEW: executes the stop trigger command sequence for the 2264 digitizers (see the 2264 User's Manual, proper station number must be set)

example: N (CR)

"COMPUTER CONTROLLED STOP"

OVERLAY: sets the overlay flag in the graphics routines, each subsequent graph will be overlaid on the present set of axes

example: O (CR) (no response from computer)

PANEL: returns the front panel switch positions on the 2264 digitizers (proper station number must be set)

example: P (CR)

"2 CHNNLS 5 μS/SMPL 3K POST TRIG"

"CH1 -1 CH2 +1" (offset switches)

R: executes an F(2) command which reads one data word at a time from the 2264 buffer memories (proper station number must be set), each data word contains two samples

example: R (CR)

"F2"

"sample 1, sample 2, data word"

V#: voltage, reads the data from the 32 channel DVM (proper station number must be set)

example: V4 (CR) # = 0 - 31

"x VOLTS" x = data value

WIDTH: sets the height and width (in number of lines and number of characters, respectively) of subsequent graphs plotted on the 4025

example: W (CR)

"ENTER HEIGHT, WIDTH"

34, 80 (CR) for a full screen graph

When one of these commands has finished executing (or if the entered command was not found in the address table) the computer prints "ENTER COMMAND" and waits for the next command. The program can be stopped by the "HALT" command which returns to the BASIC monitor. The storage routines are still in active memory, however, and can be restarted by typing:

CLOSE (CR)	to insure that all data files are closed
"READY"	
GO TO 1000 (CR)	to restart program
"ENTER COMMAND"	

Sec. 5.8 Recovery From a Crash

Depending on the severity of a crash, the storage programs sometimes be restarted without having to download them from the system disk. There are three levels of crashes: those that halt the program and return to BASIC (the computer gives the "READY" prompt), those that return to the RT-11 monitor (the computer gives the "." prompt), and those which stop the entire system (RUN light is off).

Crashes to BASIC are usually caused by some illegal combination of BASIC commands such as trying to "OPEN" a file that is already open or accessing a virtual file as a sequential file. This type of crash will also occur when the data disk becomes full. Crashes to BASIC can be recovered by the method discussed in Sec. 5.7.

Crashing to the RT-11 monitor may be the only way to exit a BASIC program that has hung in a loop. This is accomplished by typing a double "control C" (only one if program is waiting for input) and the computer responds with the "." prompt. The BASIC routines have been erased from active memory but may be recovered under some circumstances from the "SWAP.SYS"[12] file on the system disk. This is done with the RT-11 "REENTER" command and the computer should respond with the BASIC "READY" prompt. The store routines can then be restarted as described

in Sec. 5.7. If any other response is given to the "REENTER" command then the computer has entered some RT-11 program and not BASIC. Return to the RT-11 monitor by typing "control C". The BASIC program must then be reloaded as described in Sec. 5.5.

Generally, if a crash is severe enough to halt the system (RUN light out), such as trying to access nonexistent equipment (be sure CAMAC crate is ON LINE), both BASIC and the RT-11 monitor have been erased and must be reloaded (e.g. system must be rebooted). In some instances, the system can be restarted by toggling the halt/continue switch. If recovered, the computer responds with the "." prompt and BASIC can be recovered with the "REENTER" command.

Since the battery backup option was not purchased with the present system, any interruption in the computer's AC power will cause all programs in active memory to be lost. The system must then be rebooted.

CHAPTER VI

CONCLUSIONS AND FUTURE IMPROVEMENTS

A versatile, flexible, minicomputer based data acquisition and retrieval system has been designed and interfaced to the Texas Tech Tokamak. The system includes a DEC PDP-11/04 minicomputer, CAMAC digitizers, a Tektronix graphics terminal, and BASIC programs to drive the system. The software can execute all standard CAMAC commands as well as user written routines. The digitized data are stored on disks in a fixed format which allows for straightforward retrieval and analysis. An example of a data analysis program is discussed in Appendix D.

As new hardware is acquired, it can be added to the software command structure simply by adding the appropriate information to the configuration file and adding the command and controlling routines to the main calling program. The hardware need not be limited to the CAMAC equipment.

At present the software is written as a series of overlaid segments due to the limited amount of memory in the 11/04 minicomputer. When additional memory becomes available, all program segments could be resident at once. This would eliminate the delay caused when segments are overlaid.

The execution time of the graphics routines could be improved by writing these routines in assembly language and linking them into the BASIC language. These routines would then be available as callable functions as are those discussed in Sec. 4.5 and Appendix B. The Tektronix 4025 terminal can be driven by the older 4010 style commands.

These commands are in a more compressed form which is less convenient to use but considerably faster than the 4025 commands for the present application. If the graphics routines were written in 4010 style commands and then linked into BASIC, both speed and convenience would be preserved.

Some improvements and additions could be made to the user functions that have been added to the BASIC language. As a safety feature, a routine could be written which allows the user to limit the range (in memory) of the memory writing routines. This would assure that the area of memory containing the BASIC language could not be modified. Another useful routine would be one which checked to see if the hardware setup corresponded to the information in the configuration file. This might be done by executing a CAMAC command at each station where a module is supposed to be and then checking the results. Any discrepancies could be typed on the terminal.

The storage routines were written in BASIC to simplify modifications and debugging during their development. Now that the routines are somewhat finalized, they could be translated into FORTRAN. The FORTRAN language would provide a faster, more structured program with increased mathematical capabilities; but would increase the effort required to make additions to the programs.

The present data file format was designed to simplify the reading out of data. In accomplishing this, the data have been stored in less than optimal form and some redundancy has occurred. For example, the parameter file is added to each data file but may contain the same information for several shots in a row. The parameter information makes

up approximately 4.2 blocks of each data file. Substantial saving in disk space could result if some indexing scheme were incorporated which allowed the parameter file to be stored only when its information is changed. Another possibility would be to store only the numerical part and require the reading programs to extract headings and comments from a single heading on each data disk.

At present, The majority of the data stored come from the 2264 digitizers. These units produce 8 bit data words which are stored by BASIC as 16 bit words. The disk space required to store these data could be cut in half if some means of prepacking these data before they are stored were incorporated. Other methods of data compression include calculating the exact number of bits necessary for each data word or storing the differences between data points rather than the points themselves. Each of these metheds will increase the data density at the cost of increased complexity within the reading programs.

LIST OF REFERENCES

1. H. C. Kirbie, "Design and Construction of the Texas Tech Tokamak", M.S. Thesis, Dept. of Elect. Engr., Texas Tech University (1978).
2. M. Kristiansen and M. Hagler, editors, "Proceedings of the Small Toroidal Plasma Devices User's Meeting", Plasma Laboratory, Dept. of Elect. Engr., Texas Tech University (1978).
3. Institute of Electrical and Electronics Engineering, CAMAC Instrumentation and Interface Standards (John Wiley & Sons Inc., New York, 1976).
4. Ibid., p. 4.
5. PDP-11 Processor Handbook (Digital Equipment Corp., Maynard, MA, 1978), Ch. 2.
6. System User's Guide, 11/04 Documentation Kit (Digital Equipment Corp., Maynar, MA, 1977), Table 3-4.
7. Basic/RT-11 Language Reference Manual (Digital Equipment Corp., Maynard, MA, 1976), Ch. 8, App. D, App F.
8. Ibid., Sec. 5.9.
9. Introduction to RT-11, (Digital Equipment Corp., Maynard, MA, 1978), p. 2-4.
10. Tektronix 4025 Computer Display Terminal Operator's Manual, (Tektronix Inc., Beaverton, OR, 1978), p. 3-18 to 3-21.
11. Ibid., Ch. 5.
12. System User's Guide, 11/04 Documentation Kit (Digital Equipment Corp., Maynard, Ma, 1977), p. 1-2.

APPENDIX A

PROGRAM LISTINGS AND DISCUSSIONS

A listing of each of the main programs making up the storage routines is included in this section. Following each listing is a line by line discussion of the program. The programs CREATE.BAS, CATDIR.BAS, and CATDIR.COM are also included in this section. A list of subroutines used in the storage routine is included in Table A-2.

Contents

	Page
OVRNEW.BAS	86
PLT200. BAS	93
Graphics Routines Entry Points	97
RED200.BAS	98
STO200.BAS	102
GRA200.BAS	107
FIXRED.BAS	107
Subroutines Used in Storage Routines	108
CREATE.BAS	109
CATDIR.BAS	112
CATDIR.COM	114

OVRNEW.BAS

OVRNEW.BAS is the main calling program and is downloaded and run by BASIC language commands. Lines 10 to 20045 make up the OVRNEW.BAS program. This program contains most of the system command routines and the logic for overlaying other program segments. Lines 20050 to 20990 initially contain the PLT200.BAS program and will later be used for other segments as required. The OVRNEW.BAS program is discussed in Sec. 4.6.

LINES	PURPOSE
10	set C$ = ! as the 4025 command character (see the 4025 User's Manual) otherwise, 4025 commands would execute when the program is listed
	set B$ = "BELL"; set constants to zero
110	check system date and time,set E1 = 1 if date not set or time not after 4:00 A.M.
300 to 330	check INITIA.LIZ file on data disk to determine if this is new start or restart, if restart ask operator to verify shot number
400 to 450	convert date to decimal format,e.g. YYMMDD, store in variable D2$
500 to 540	print hardware configuration file into workspace on 4025 terminal
900 to 950	dimension arrays and set default constants
960 to 980	print rerun message and erase lines 10 to 980
990 to 995	prompt operator for information and set

	CAMAC CSR address (by subroutine 2905)
1000	reentry point common to all overlays
	enter command from operator
1010 to 1030	branch address table for three letter commands
1045 to 1093	branch address table for one letter commands
1150 to 1181	execute STATUS command
1210	execute A (advance) command
1250	start execution of SUBADDRESS command
1260	execute DATAPOINTS command
1300 to 1310	execute the READ command (single data point from 2264 digitizer)
1350	execute SHOT command
1370	execute NAME command
1400 to 1410	execute the AUT (auto trigger) command
1450	execute OVERLAY command (set OVERLAY flag)
1470	execute MULTIPLE command (set MULTIPLE flag)
1480	execute WIDTH command
1500	execute CAMAC F code
1600	execute ERASE command
1650	execute SCALE command
1700 to 1765	execute PANEL command
1800 to 1830	execute V (2232 voltage) command
1900 to 1980	execute PARAMETER command; print parameter file into workspace; put 4025 in form fillout mode (see 4025 User's Manual) and allow operator to modify and save parameter file

2000 to 2005	subroutine to separate upper and lower bytes of data word returned by PEEK function
2500 to 2503	load 16 bit CSR into array B(1) to B(16)
2900 to 2907	execute CRATE command
2912	subroutine to calculate total CAMAC address from crate number, station number, and sub-address (see DCC-11 User's Manual)
3000 to 3060	define functions used by storage routines
5000	subroutine to reset (cycle through) the 2264 buffer memory
6000 to 6140	execute NEW command
7000 to 8000	subroutine for executing CAMAC F codes using PEEK and POKE, etc. (entry point is 7900)
9000 to 9050	execute INITIALIZE command
9200	subroutine to prepare workspace to be sent to the computer starting at line S1
9300	subroutine to reset terminal after computer has read workspace
9400 to 9410	subroutine to set F3$ = (first twenty characters of first line of workspace)
9500 to 9520	subroutine to check time and store in variable T1 in seconds after midnight
9600 to 9650	execute TITLE command
20005 to 20045	logic to determine which overlay is present and which needs to be overlaid
20050 to 20990	contain the PLT200.BAS program

65000 to 65020	subroutine to "dump" a sequential file on the terminal (e.g. no options), entered by the BASIC GO TO 65000 command

```
02-JAN-80 COPY OF OVRNEW.BAS, OPTION I     LPRINT V1:BB

10 C$=CHR$(33)\B2$=CHR$(7)\S5=0\E1=0
100 REM     CHECK DATE
110 IF DAT$<>'           ' THEN 200 \E1=1\PRINT B2$\PRINT 'SET DATE xxxx'
200 REM     CHECK TIME
210 CALL 'PEEK'(64,D)\CALL 'PEEK'(D+2,A6)\A6=A6+D+2\GOSUB 9500
220 IF T1>14400 THEN 230 \E1=1\PRINT B1$\PRINT 'SET TIME xxxx'
230 IF E1<>1 THEN 300 \PRINT 'GO BACK TO RT-11'\CRASH E1=1 IS ERROR CODE
300 REM OPEN INIT. FILE TO CHECK IF NEW START OR RE-START AND SET SHOT #
310 OPEN 'INITIA.LIZ' AS FILE VF1$(2)=16\IF VF1(1)<>DAT$ THEN 330
320 PRINT 'LAST SHOT# = ';VF1(2),'ENTER TRUE LAST SHOT#';\INPUT S5
330 VF1(1)=DAT$\VF1(2)=STR$(S5)\CLOSE VF1
400 I5=1\REM CONVERT DATE TO DECIMAL  YY.MMDD
410 N3=POS(DAT$,'-',1)
420 READ L3$\IF L3$=SEG$(DAT$,N3+1,N3+3) THEN 430 \I5=I5+1\GO TO 420
430 D3=VAL(SEG$(DAT$,8,9))+I5/100+VAL(SEG$(DAT$,1,2))/10000
440 D2$=SEG$(STR$(D3),1,2)&SEG$(STR$(D3)&'00',4,7)\D4$=D2$
450 DATA 'JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC'
500 OPEN 'PGM:CONFIG.INF' FOR INPUT AS FILE #1\INPUT #1:L3$\FOR I1=1 TO 4
510 INPUT #1:N3,Q1(I1),Q2(I1),Q3(I1),Q4(I1),Q5(I1),Q6(I1)
520 NEXT I1\RESTORE #1\PRINT C$'WOR 20 H'
530 INPUT #1:L3$\PRINT L3$\IF END #1 THEN 540 \GO TO 530
540 PRINT C$'MON H'\CLOSE #1
900 DIM Q1(4),Q2(4),Q3(4),Q4(4),Q5(4),Q6(4)
910 DIM B(16)\O8=0\N8=0\Z5=100\Y3=-130\Y5=130\C8=2\S8=6\S2=70
920 DIM R4(10),N6(10),C3(10),U1(10),G1(10),O1(10),T(10),F1(10),N2(10)
930 DIM E1(10),S1(10),L6(11),C1(10),I1(10)\O2$='GRA'\B1$='          '
940 F7=1\F8=1\F9=1\PRINT C$;'LIN 1'\O8=3\P8=7\X$='TIME'\S3=100000
950 Y$='DIGT. UNITS'\X4=0\X6=500\Y6=259\Y4=0\F2$='NO PARMS'\GOSUB 2912 \PRINT
960 PRINT \PRINT 'DO NOT USE THE RUN COMMAND TO REENTER THIS PROGRAM'
970 PRINT 'IF PROGRAM SHOULD STOP OR CRASH USE:  "GO TO 1000" TO REENTER'
980 PRINT \PRINT \PRINT 'WAIT PLEASE'\OVERLAY 'PGM:CLRLOW'\PRINT \PRINT B2$
990 PRINT 'ENTER PARAMETER FILE NAME';\INPUT P1$\P1$='PGM:'&P1$
995 PRINT "ENTER OPERATOR'S NAME-";\INPUT N1$\GO TO 2905
1000 PRINT B2$\PRINT "ENTER COMMAND"
1010 INPUT C1$\C1$=TRM$(C1$)\L=LEN(C1$)\F$=SEG$(C1$,1,3)\IF F$="AUT" THEN 1400
1015 F$=SEG$(C1$,1,3)\IF F$='STA' THEN 1150 \IF F$='SCA' THEN 1650
1020 IF F$='SUB' THEN 1250 \IF F$='STO' THEN 20020 \IF F$='ERA' THEN 1600
1025 IF F$='REA' THEN 20035 \IF F$='PAR' THEN 1900 \IF F$='SHO' THEN 1350
1030 IF F$='NAM' THEN 1370 \IF F$='TIT' THEN 9600
1045 F$=SEG$(C1$,1,1)
1050 IF F$="I" THEN 9000 \IF F$="F" THEN 1500 \IF F$="H" THEN 1100
1065 IF F$="E" THEN 2905 \IF F$="C" THEN 1200 \IF F$="P" THEN 1700
1080 IF F$="R" THEN 1300 \IF F$="V" THEN 1800 \IF F$="G" THEN 20005
1090 IF F$="N" THEN 6000 \IF F$="O" THEN 1450 \IF F$="M" THEN 1470
1093 IF F$="W" THEN 1480 \IF F$="D" THEN 1260 \IF F$='A' THEN 1210 \GO TO 1000
1100 STOP
1150 CALL "PEEK"(A8,N)\GOSUB 2500
1170 PRINT "LAM  Q  X  F3 F2 F1 = OCTAL 13 12 11 10  9  8  7  6  5  4  3  2  1"
1180 PRINT B(10);' ';B(8);B(7);" ";B(3);B(2);B(1);"   ";4*B(3)+2*B(2)+B(1);
1181 PRINT "  ";\FOR I1=13 TO 1 STEP -1\PRINT B(I1);\NEXT I1\PRINT \GO TO 1010
1200 GO TO 2900
1210 N4=INT(VAL(SEG$(C1$,2,L)))\CALL 'FAST'(A9,0,N4,0,Q,D)\GO TO 1000
1250 PRINT "ENTER SUBADDRESS"\INPUT S7\GO TO 2903
1260 PRINT '# OF DATA POINTS'\INPUT Z5\GO TO 1000
1300 CALL 'FAST'(A9,0,1,256,Q,D1)\D=D1\GOSUB 2000 \PRINT U1;L1;D1
1310 IF Q=1 THEN 1010 \PRINT CHR$(7)\GO TO 1000
1350 PRINT 'LAST SHOT# =';S5\PRINT 'ENTER LAST SHOT#';\INPUT S5\GO TO 1000
1370 PRINT "ENTER OPERATOR'S NAME";\INPUT N1$\GO TO 1000
1400 N4=VAL(SEG$(C1$,4,L))\FOR I1=1 TO N4\Z=21\Z=SIN(Z)+COS(Z)\F=9\GOSUB 7900
1410 F=25\GOSUB 7900 \NEXT I1\GO TO 1000
1450 O8=1\GO TO 1010
1470 O8=0\M8=1\GO TO 1010
1480 PRINT 'ENTER WIDTH,HEIGHT'\INPUT X6,Y6\X6=X6*8\Y6=Y6*14\GO TO 1010
1500 F=VAL(SEG$(C1$,2,L))\GOSUB 7900 \PRINT "F";F\PRINT D\GO TO 1010
1600 O8=0\M8=0\P8=7\Q8=3\GO TO 1010
1650 PRINT 'ENTER YMIN,YMAX'\INPUT Y3,Y5\GO TO 1010
1700 F=1\GOSUB 7900 \N=D\GOSUB 2500 \IF B(10)=0 THEN 1740 \PRINT "RATE TOO HI"
1740 IF B(9)=1 THEN 1750 \IF FNC(X)<>1 THEN 1750 \PRINT 'ODD FLAG'
1750 PRINT FNC(X);"CHANNELS";FNT(X);"U SEC/SAMPLE";FNP(X);"K POST TRIG"
1752 N8=FNC(X)\F=0\GOSUB 7900 \N=D\GOSUB 2500 \FOR I1=1 TO N8
1765 PRINT "/CH";I1;' ';B(2*I1-1)-B(2*I1);\NEXT I1\PRINT \GO TO 1010
1800 S7=VAL(SEG$(C1$,2,L))\F=INT(S7/16)\S7=S7-F*16
1830 GOSUB 2912 \GOSUB 7900 \V=D/4096*10-5\PRINT V;"VOLTS"\GO TO 1010
1900 PRINT 'PARAMETER FILE';\INPUT L3$\IF L3$='W' THEN 1940 \P1$='PGM:'&L3$
1910 PRINT C$'WOR 20 H'\OPEN P1$ AS FILE #1\L2=0
1920 INPUT #1:L3$\PRINT L3$\L2=L2+1\IF END #1 THEN 1930 \GO TO 1920
1930 PRINT C$'MON H'\CLOSE #1\GOSUB 9400 \F2$=F3$
1940 PRINT C$'FOR Y'\PRINT C$'JUM'\PRINT 'MAKE CHANGES NOW THEN ANSWER (Y\N)'
1950 PRINT 'SAVE NEW FILE';\INPUT L3$\PRINT C$'FOR N'
1960 IF SEG$(L3$,1,1)<>'Y' THEN 1000 \PRINT 'NEW NAME';\INPUT P1$\P1$='PGM:'&P1$
```

```
1970 OPEN P1$ FOR OUTPUT(10) AS FILE #1\S1=1\GOSUB 9200 \FOR I1=1 TO L2
1980 INPUT L3$\PRINT #1:L3$\NEXT I1\GOSUB 9300 \CLOSE #1\GO TO 1000
2000 IF D>=0 THEN 2005 \D=D+65536
2005 U1=INT(D/256)\L1=127-D+256*U1\U1=127-U1\RETURN
2500 IF N>=0 THEN 2503 \N=N+65536
2503 FOR I1=1 TO 16\B(I1)=N-2*INT(N/2)\N=INT(N/2)\NEXT I1\RETURN
2900 PRINT 'ENTER CRATE #,SLOT #'\INPUT C8,S8\S7=0
2903 GOSUB 2912
2905 PRINT 'CRATE';C8;'SLOT';S8;'SUBADDRESS';S7;'AT';A9;'    CRATE CSR AT';A8
2907 GO TO 1000
2912 B1=-7168\A9=INT(B1+1024*C8+32*S8+S7*2)\A8=B1+1024*C8+768\RETURN
3000 DEF FNP(X)=8-4*B(3)-2*B(2)-B(1)
3010 DEF FNT(X)=5*(1-B(4)/2)*10^(2-2*B(6)-B(5))
3020 DEF FNC(X)=2^(3-2*B(8)-B(7))
3050 DEF FNA(X)=INT((X-1)/2)
3060 DEF FNQ(X,Y)=SEG$(B1$,LEN(STR$(X)),Y)&STR$(X)&','
5000 CALL 'FAST'(A9,0,17000,0,Q,D)\RETURN
6000 PRINT 'COMPUTER CONTROLLED STOP'\F=26\GOSUB 7900 \F=25\GOSUB 7900 \F=10
6140 GOSUB 7900 \F=16\GOSUB 7900 \GO TO 1000
7000 N3=F-8*INT(F/8)\CALL 'POKE'(A8,N3)\IF F>15 THEN 7100
7020 CALL 'PEEK'(A9,D)\RETURN
7100 CALL 'POKE'(A9,0)\PRINT 'F';F\RETURN
7900 IF A8*A9<>0 THEN 7910 \PRINT 'SET CRATE #,SLOT #'\GO TO 1000
7910 N3=8*INT(F/8)\N3=N3/8-2*INT(N3/16)\IF N3=1 THEN 8000 \GO TO 7000
8000 CALL 'STOR'(A9,F)\RETURN
9000 PRINT 'INITIALIZE? Y/N'\INPUT L3$\IF L3$='Y' THEN 9050 \GO TO 1010
9050 CALL 'PEEK'(-5118,N3)\GO TO 1010 \STOP
9200 PRINT C$'PRO 10/?/'C$'BUF'C$'SEN'\FOR I5=1 TO S1-1\INPUT L3$\NEXT I5\RETURN
9300 PRINT C$'REP 00'\FOR I5=1 TO 150\INPUT L3$
9310 IF SEG$(L3$,1,4)=C$&'ANS' THEN 9320 \NEXT I5\PRINT 'NO RPLY FROM 4025'\CRSH
9320 PRINT C$'BUF N'\PRINT C$'PRO'\PRINT 'END WS'\RETURN
9400 F3$=''\FOR I5=1 TO 20\PRINT C$'JUM 1';I5\PRINT C$'REP 01'
9410 PRINT C$'UP 1'\INPUT L3$\F3$=F3$&SEG$(L3$,17,17)\NEXT I5\RETURN
9500 CALL 'PEEK'(A6,H)\IF H>=0 THEN 9510 \H=H+2^16
9510 CALL 'PEEK'(A6+2,L)\IF L>=0 THEN 9520 \L=L+2^16
9520 T1=INT(L/60+H*2^16/60)\RETURN
9600 OPEN 'PGM:TITLE.REC' AS FILE VF3%=16\PRINT 'TYPE 'STOP' TO EXIT PROGRAM'
9603 PRINT ''TYPE' IS A 3 DIGIT NUMBER, 'TITLE' MUST BE <=12 CHARS. LONG'
9605 PRINT 'ENTER TYP,TITLE ';\INPUT L3$\IF L3$='STOP' THEN 9650
9610 FOR I1=1 TO 999\IF SEG$(VF3(I1),5,7)='END' THEN 9630
9620 IF VAL(SEG$(L3$,1,3))=VAL(SEG$(VF3(I1),1,3)) THEN 9640 \NEXT I1
9630 VF3(I1)=L3$\VF3(I1+1)='999END'\GO TO 9605
9640 PRINT 'OLD TYPE ';VF3(I1)\VF3(I1)=L3$\PRINT 'NEW TYPE ';VF3(I1)\GO TO 9605
9650 CLOSE VF3\GO TO 1000
20005 IF O2$='GRA' THEN 20050 \IF O2$='STO' THEN 20008
20006 PRINT 'WAIT (GRASHORT)'\OVERLAY 'PGM:GRA200'
20007 GO TO 20010
20008 PRINT 'WAIT (GRA)'\OVERLAY 'PGM:PLT200'
20010 O2$='GRA'\GO TO 20050
20020 IF O2$='STO' THEN 20050
20025 PRINT 'WAIT (STORE)'\OVERLAY 'PGM:STO200'
20030 O2$='STO'\GO TO 20050
20035 IF O2$='RED' THEN 20050 \IF O2$='STO' THEN 20040
20036 PRINT 'WAIT (READS)'\OVERLAY 'PGM:FIXRED'
20037 GO TO 20045
20040 PRINT 'WAIT (READ)'\OVERLAY 'PGM:RED200'
20045 O2$='RED'
20050 GOSUB 5000 \PRINT 'CHANNEL'\INPUT C3
20060 Y$='CHANNEL '&STR$(C3)\F7=1\F8=1\F9=1\IF M8=0 THEN 20090
20070 P8=P9+10\F7=1\F8=0\F9=0\IF P8+INT((X6-X4)/8)+1<80 THEN 20090
20080 P8=7\Q8=Q9+5
20090 F=1\GOSUB 7900
20100 N=D\GOSUB 2500 \N8=FNC(X)\IF C3>N8 THEN 20210
20110 PRINT '# SMPLS? (MULTIPLES OF';Z5;'), 1ST SMPL #?'\INPUT H4,A3\A3=INT(A3)
20120 IF H4<Z5 THEN 20220 \F=26\GOSUB 7900 \F=10\GOSUB 7900
20130 S7=FNA(C3)\GOSUB 2912 \F=16\GOSUB 7900 \CALL 'FAST'(A9,0,A3,0,Q,D)
20140 U2=INT(1-2*(C3-2*INT(C3/2)))\X3=A3\X5=A3+H4\GOSUB 20650 \Q4=0
20150 IF O8=1 THEN 20160 \GOSUB 20250
20160 W9=0\H5=INT(H4/Z5)\X=A3\FOR I1=0 TO Z5\CALL 'FAST'(A9,U2,H5,256,Q,Y)
20170 GOSUB 20560 \X=X+H5\NEXT I1\IF O8=1 THEN 20190 \GOSUB 20540
20190 GOSUB 5000 \S7=0\F=24\GOSUB 7900 \S7=FNA(C3)\F=16\GOSUB 7900
20200 GO TO 1000
20210 PRINT 'CHNNL NOT SMPLD'\GO TO 1000
20220 PRINT 'NOT ENOUGH'\GO TO 20110
20230 REM
20240 REM
20250 Q9=Q8+INT(Y6/14)\P9=P8+INT(X6/8)\IF Q8<0 THEN 20270
20260 IF Q8>Q9 THEN 20270 \GO TO 20280
20270 PRINT 'ERR VRT. SIZE OR POS. Q8=';Q8;'Q9=';Q9\STOP
20280 IF P9>80 THEN 20300 \IF P8<1 THEN 20300 \IF P9<P8 THEN 20300
20290 GO TO 20310
20300 PRINT 'ERR HORZ. SIZE OR POS. P8=';P8;'P9=';P9\STOP
```

```
20310 IF F9=0 THEN 20340 \IF Q9+3<31 THEN 20330 \PRINT C$;'WOR 20'
20320 PRINT C$;'MON H;ERR WS>30';CHR$(7)\GO TO 20340
20330 PRINT C$;'WOR';Q9+3
20340 IF F8=0 THEN 20350 \PRINT C$;'ERA W'
20350 PRINT C$;'GRA ';Q8;Q9;P8;P9\P7=P8-1
20360 IF (P9-P8+1)*(Q9-Q8+1)<2200 THEN 20380
20370 PRINT CHR$(7);'GTB?'
20380 RETURN
20390 PRINT C$;'WOR H'\PRINT C$;'JUMP ';Q9+3;INT((P8+P9-LEN(X$))/2)
20400 PRINT X$;\IF P7>1 THEN 20410 \P7=2
20410 FOR I9=1 TO LEN(Y$)\PRINT C$;'JUM ';INT((Q8+Q9-LEN(Y$))/2)+I9;P7-1
20420 PRINT SEG$(Y$,I9,I9)\NEXT I9\P7=P7-2\PRINT C$;'MON H'\RETURN
20430 PRINT C$;'WOR H'\P1=P4\Q1=Q4\Q3=0\IF X7=1 THEN 20470
20440 P2=10^INT(LOG(ABS(X5-X3))/LOG(10))\P2=P2/10\P3=1\GOSUB 20760 \P2=10*P2
20450 IF P2<.5*ABS(X5-X3) THEN 20460 \P2=P2/2
20460 P3=-2\GOSUB 20760 \GO TO 20480
20470 P2=10\P3=-2\GOSUB 20760 \P3=1\FOR I7=2 TO 9\P1=I7*P4\GOSUB 20760 \NEXT I7
20480 P3=0\IF Y7=1 THEN 20520 \Q2=10^INT(LOG(ABS(Y5-Y3))/LOG(10))
20490 Q2=Q2/10\Q3=1\GOSUB 20760 \Q2=10*Q2
20500 IF Q2<.5*ABS(Y5-Y3) THEN 20510 \Q2=Q2/2
20510 Q3=-2\GOSUB 20760 \GO TO 20530
20520 Q2=10\Q3=-2\GOSUB 20760 \Q3=1\FOR I7=2 TO 9\Q1=I7*Q4\GOSUB 20760 \NEXT I7
20530 RETURN
20540 GOSUB 20740 \GOSUB 20430 \GOSUB 20390 \RETURN
20550 W9=2
20560 IF X7=1 THEN 20570 \X1=INT((X-X3)*X8+X4+.5)\GO TO 20580
20570 X1=INT(LOG(X/X3)*X8+X4+.5)
20580 IF Y7=1 THEN 20600
20590 Y1=INT((Y-Y3)*Y8+Y4+.5)\GO TO 20610
20600 Y1=INT(LOG(Y/Y3)*Y8+.5)
20610 IF W9=1 THEN 20640 \IF W9=2 THEN 20630 \W9=1
20620 X2=X1\Y2=Y1
20630 RETURN
20640 PRINT C$;'VEC';X2;Y2;X1;Y1\GO TO 20620
20650 P4=X3\Q4=Y3\IF X7=1 THEN 20660 \X8=(X6-X4)/(X5-X3)\GO TO 20690
20660 IF X5*X3<>0 THEN 20680
20670 PRINT C$;'MON H;LOG ERR L X=';X3;'R X=';X5\STOP
20680 X8=(X6-X4)/LOG(X5/X3)
20690 IF Y7=1 THEN 20700 \Y8=(Y6-Y4)/(Y5-Y3)\GO TO 20730
20700 IF Y3*Y5<>0 THEN 20720
20710 PRINT C$;'MON H;LOG ERR BOTH Y=';Y3;'TOP Y=';Y5\STOP
20720 Y8=(Y6-Y4)/LOG(Y5/Y3)
20730 RETURN
20740 X=P4\Y=Q4\GOSUB 20550 \PRINT C$;'VEC';X4;Y1;X6;Y1
20750 PRINT C$;'VEC';X1;Y4;X1;Y6\RETURN
20760 PRINT C$'WOR H'\P5=INT(((ABS(Y6-Y4)/100)+2)*P3)
20770 Q5=INT(((ABS(X6-X4)/100)+2)*Q3)\IF P5=0 THEN 20870 \Y=Q4
20780 FOR I9=0 TO 800\IF X7=1 THEN 20790 \X=P1+I9*P2\GO TO 20800
20790 X=P1*(P2^I9)
20800 GOSUB 20550 \IF X1>X6 THEN 20810 \IF X1<X4 THEN 20810 \NEXT I9
20810 FOR I8=I9-1 TO -800 STEP -1\IF X7=1 THEN 20820 \X=P1+I8*P2\GO TO 20830
20820 X=P1*(P2^I8)
20830 GOSUB 20550 \IF X1>X6 THEN 20870 \IF X1<X4 THEN 20870
20840 PRINT C$;'VEC';X1;Y1;X1;Y1+P5\PRINT C$;'VEC';X1;Y1;X1;Y1-P5
20850 IF P5>0 THEN 20860 \PRINT C$;'JUMP 'Q9+1;P8+INT((X1-12)/8)\PRINT X;
20860 NEXT I8
20870 IF Q5=0 THEN 20990 \X=P4\FOR I9=0 TO 800\IF Y7=1 THEN 20890
20880 Y=Q1+I9*Q2\GO TO 20900
20890 Y=Q1*(Q2^I9)
20900 GOSUB 20550 \IF Y1>Y6 THEN 20910 \IF Y1<Y4 THEN 20910 \NEXT I9
20910 FOR I8=I9-1 TO -800 STEP -1\IF Y7=1 THEN 20930
20920 Y=Q1+I8*Q2\GO TO 20940
20930 Y=Q1*(Q2^I8)
20940 GOSUB 20550 \IF Y1>Y6 THEN 20990 \IF Y1<Y4 THEN 20990
20950 PRINT C$;'VEC';X1;Y1;X1+Q5;Y1\PRINT C$;'VEC';X1;Y1;X1-Q5;Y1
20960 IF Q5>0 THEN 20980
20965 PRINT C$;'JUMP ';Q9-INT(Y1/14);P8-LEN(STR$(Y))\PRINT STR$(Y)
20970 IF P8-LEN(STR$(Y))-1>P7 THEN 20980 \P7=P8-LEN(STR$(Y))-1
20980 NEXT I8\PRINT C$;'MON H'
20990 RETURN
65000 PRINT 'FILE';\INPUT L3$\OPEN L3$ AS FILE #7
65010 INPUT #7:L3$\PRINT L3$\IF END #7 THEN 65020 \GO TO 65010
65020 CLOSE #7\PRINT \PRINT 'END OF FILE'\GO TO 1000
```

PLT200.BAS

The PLT200.BAS program is overlaid in response to the GRAPH command. Lines 20050 to 20240 are identical to GRA200.BAS and serve to extract data points from the 2264 buffer memory and plot them on the 4025 terminal. Lines 20250 to 20990 contain a set of general purpose graphing routines to drive the 4025 graphics terminal. These routines, called EASYG2.BAS, normally reside in lines 17000 to 19700 but have been resequenced to correspond with the overlay area in the storage routines. The new subroutine addresses are shown in Table A-1. The PLT200.BAS program is discussed in Sec. 4.8.

LINES	PURPOSE
20050	reset 2284 buffer; input channel to be plotted
20060 to 20080	reset graphics parameters;
20090 to 20100	determine how many channels were sampled
20110	enter number of points to plot and starting point (in data points)
20120	check that number of points is greater than the number of points in graph (set by the DATA command); abort if not; start the 2264 readout sequence (F26, F10)
20130	set subaddress; continue readout sequence (F16, FAST)
20140 to 20150	determine high or low byte; set graphics parameters
20160 to 20170	read and plot each point

20190	reset 2264 readout (F24, F16) to enable monitor display
20200	return for next command
20210 to 20220	error messages
20250 to 20990	plotting routines

```
02-JAN-80 COPY OF PLT200.BAS, OPTION I    LPRINT V1:BB

20050 GOSUB 5000 \PRINT B2$\PRINT 'CHANNEL';\INPUT C3
20060 Y$='CHANNEL '$STR$(C3)\F7=1\F8=1\F9=1\IF M8=0 THEN 20090
20070 P8=P9+10\F7=1\F8=0\F9=0\IF P8+INT((X6-X4)/8)+1<80 THEN 20090
20080 P8=7\Q8=Q9+5
20090 F=1\GOSUB 7900
20100 N=D\GOSUB 2500 \N8=FNC(X)\IF C3>N8 THEN 20210
20110 PRINT '# SMPLS? (MULTIPLES OF';Z5;'), 1ST SMPL #?'\INPUT H4,A3\A3=INT(A3)
20120 IF H4<Z5 THEN 20220 \F=26\GOSUB 7900 \F=10\GOSUB 7900
20130 S7=FNA(C3)\GOSUB 2912 \F=16\GOSUB 7900 \CALL 'FAST'(A9,0,A3,0,Q,D)
20140 U2=INT(1-2*(C3-2*INT(C3/2)))\X3=A3\X5=A3+H4\GOSUB 20650 \Q4=0
20150 IF Q8=1 THEN 20160 \GOSUB 20250
20160 W9=0\H5=INT(H4/Z5)\X=A3\FOR I1=0 TO Z5\CALL 'FAST'(A9,U2,H5,256,Q,Y)
20170 GOSUB 20560 \X=X+H5\NEXT I1\IF Q8=1 THEN 20190 \GOSUB 20540
20180 REM
20190 GOSUB 5000 \S7=0\F=24\GOSUB 7900 \S7=FNA(C3)\F=16\GOSUB 7900
20200 GO TO 1000
20210 PRINT 'CHNNL NOT SMPLD'\GO TO 1000
20220 PRINT 'NOT ENOUGH'\GO TO 20110
20230 REM
20240 REM
20250 Q9=Q8+INT(Y6/14)\P9=P8+INT(X6/8)\IF Q8<0 THEN 20270
20260 IF Q8>Q9 THEN 20270 \GO TO 20280
20270 PRINT 'ERR VRT. SIZE OR POS. Q8=';Q8;'Q9=';Q9\STOP
20280 IF P9>80 THEN 20300 \IF P8<1 THEN 20300 \IF P9<P8 THEN 20300
20290 GO TO 20310
20300 PRINT 'ERR HORZ. SIZE OR POS. P8=';P8;'P9=';P9\STOP
20310 IF F9=0 THEN 20340 \IF Q9+3<31 THEN 20330 \PRINT C$;'WOR 20'
20320 PRINT C$;'MON H;ERR WS>30';CHR$(7)\GO TO 20340
20330 PRINT C$;'WOR';Q9+3
20340 IF F8=0 THEN 20350 \PRINT C$;'ERA W'
20350 PRINT C$;'GRA ';Q8;Q9;P8;P9\P7=P8-1
20360 IF (P9-P8+1)*(Q9-Q8+1)<2200 THEN 20380
20370 PRINT CHR$(7);'GTB?'
20380 RETURN
20390 PRINT C$;'WOR H'\PRINT C$;'JUMP ';Q9+3;INT((P8+P9-LEN(X$))/2)
20400 PRINT X$;\IF P7>1 THEN 20410 \P7=2
20410 FOR I9=1 TO LEN(Y$)\PRINT C$;'JUM ';INT((Q8+Q9-LEN(Y$))/2)+I9;P7-1
20420 PRINT SEG$(Y$,I9,I9)\NEXT I9\P7=P7-2\PRINT C$;'MON H'\RETURN .
20430 PRINT C$;'WOR H'\P1=P4\Q1=Q4\Q3=0\IF X7=1 THEN 20470
20440 P2=10^INT(LOG(ABS(X5-X3))/LOG(10))\P2=P2/10\P3=1\GOSUB 20760 \P2=10*P2
20450 IF P2<.5*ABS(X5-X3) THEN 20460 \P2=P2/2
20460 P3=-2\GOSUB 20760 \GO TO 20480
20470 P2=10\P3=-2\GOSUB 20760 \P3=1\FOR I7=2 TO 9\P1=I7*P4\GOSUB 20760 \NEXT I7
20480 P3=0\IF Y7=1 THEN 20520 \Q2=10^INT(LOG(ABS(Y5-Y3))/LOG(10))
20490 Q2=Q2/10\Q3=1\GOSUB 20760 \Q2=10*Q2
20500 IF Q2<.5*ABS(Y5-Y3) THEN 20510 \Q2=Q2/2
20510 Q3=-2\GOSUB 20760 \GO TO 20530
20520 Q2=10\Q3=-2\GOSUB 20760 \Q3=1\FOR I7=2 TO 9\Q1=I7*Q4\GOSUB 20760 \NEXT I7
20530 RETURN
20540 GOSUB 20740 \GOSUB 20430 \GOSUB 20390 \RETURN
20550 W9=2
20560 IF X7=1 THEN 20570 \X1=INT((X-X3)*X8+X4+.5)\GO TO 20580
20570 X1=INT(LOG(X/X3)*X8+X4+.5)
20580 IF Y7=1 THEN 20600
20590 Y1=INT((Y-Y3)*Y8+Y4+.5)\GO TO 20610
20600 Y1=INT(LOG(Y/Y3)*Y8+.5)
20610 IF W9=1 THEN 20640 \IF W9=2 THEN 20630 \W9=1
20620 X2=X1\Y2=Y1
20630 RETURN
20640 PRINT C$;'VEC';X2;Y2;X1;Y1\GO TO 20620
20650 P4=X3\Q4=Y3\IF X7=1 THEN 20660 \X8=(X6-X4)/(X5-X3)\GO TO 20690
20660 IF X5*X3<>0 THEN 20680
20670 PRINT C$;'MON H;LOG ERR L X=';X3;'R X=';X5\STOP
20680 X8=(X6-X4)/LOG(X5/X3)
20690 IF Y7=1 THEN 20700 \Y8=(Y6-Y4)/(Y5-Y3)\GO TO 20730
20700 IF Y3*Y5<>0 THEN 20720
20710 PRINT C$;'MON H;LOG ERR BOTH Y=';Y3;'TOP Y=';Y5\STOP
20720 Y8=(Y6-Y4)/LOG(Y5/Y3)
20730 RETURN
20740 X=P4\Y=Q4\GOSUB 20550 \PRINT C$;'VEC';X4;Y1;X6;Y1
20750 PRINT C$;'VEC';X1;Y4;X1;Y6\RETURN
20760 PRINT C$'WOR H'\P5=INT(((ABS(Y6-Y4)/100)+2)*P3)
20770 Q5=INT(((ABS(X6-X4)/100)+2)*Q3)\IF P5=0 THEN 20870 \Y=Q4
20780 FOR I9=0 TO 800\IF X7=1 THEN 20790 \X=P1+I9*P2\GO TO 20800
20790 X=P1*(P2^I9)
20800 GOSUB 20550 \IF X1>X6 THEN 20810 \IF X1<X4 THEN 20810 \NEXT I9
20810 FOR I8=I9-1 TO -800 STEP -1\IF X7=1 THEN 20820 \X=P1+I8*P2\GO TO 20830
20820 X=P1*(P2^I8)
20830 GOSUB 20550 \IF X1>X6 THEN 20870 \IF X1<X4 THEN 20870
20840 PRINT C$;'VEC';X1;Y1;X1;Y1+P5\PRINT C$;'VEC';X1;Y1;X1;Y1-P5
```

```
20850 IF P5>0 THEN 20860 \PRINT C$;'JUMP '09+1;P8+INT((X1-12)/8)\PRINT X;
20860 NEXT I8
20870 IF Q5=0 THEN 20990 \X=P4\FOR I9=0 TO 800\IF Y7=1 THEN 20890
20880 Y=Q1+I9*Q2\GO TO 20900
20890 Y=Q1*(Q2^I9)
20900 GOSUB 20550 \IF Y1>Y6 THEN 20910 \IF Y1<Y4 THEN 20910 \NEXT I9
20910 FOR I8=I9-1 TO -800 STEP -1\IF Y7=1 THEN 20930
20920 Y=Q1+I8*Q2\GO TO 20940
20930 Y=Q1*(Q2^I8)
20940 GOSUB 20550 \IF Y1>Y6 THEN 20990 \IF Y1<Y4 THEN 20990
20950 PRINT C$;'VEC';X1;Y1;X1+Q5;Y1\PRINT C$;'VEC';X1;Y1;X1-Q5;Y1
20960 IF Q5>0 THEN 20980
20965 PRINT C$;'JUMP ';Q9-INT(Y1/14);P8-LEN(STR$(Y))\PRINT STR$(Y)
20970 IF P8-LEN(STR$(Y))-1>P7 THEN 20980 \P7=P8-LEN(STR$(Y))-1
20980 NEXT I8\PRINT C$;'MON H'
20990 RETURN
```

TABLE A-1

Graphics Routines Entry Points

User Entry Points

Old Entry Point	New Entry Point	Purpose
17000	-	set graphics parameters (this is done in line 940 of OVRNEW.BAS)
18100	20250	set size of graphics area
18900	20540	draw ticks, scales (log or linear), and axes
19200	20650	set scale factors

Internal Entry Points

Old Entry	New Entry
18300	20390
18400	20430
19000	20550
19005	20560
19300	20740
19400	20760

RED200.BAS

The RED200.BAS program is overlaid in response to the READ command. Lines 20050 to 20240 are identical to FIXRED.BAS and serve to extract points from files on the data disk and plot them on the 4025 graphics terminal. Lines 20250 to 20990 are the same plotting routines discussed under PLT200.BAS. The RED200.BAS program is discussed in Sec. 4.12.

This program is intended as a rough check to see if data is being recorded properly. A program to read and analyze data is discussed in Appendix D.

LINES	PURPOSE
20050	input data file name and record type to be plotted
20060 to 20070	open data file as sequential file (see BASIC Manual), determine number of records in file
20080 to 20090	locate desired file type; print error message if not found
20100	find header address and record title
20110	go to record header line and read information
20120 to 20140	close data file; set graphics parameters
20150	check number of data points; reopen data file as virtual file (see BASIC Manual)
20160 to 20180	plot data; any system command options (MULTIPLE, OVERLAY, etc.) must have been set before entering the READ command

20190 to 20200	type directory section of data header (lines 1001 to 1017 in the example in Fig. 4-7); return to OVRNEW.BAS for next command
20210 to 20220	error messages

```
20050 PRINT B2$\PRINT 'DATA FILE'\INPUT D3$\PRINT 'RECORD TYPE'\INPUT T4
20060 OPEN D3$ FOR INPUT AS FILE #3\INPUT #3:L3$\INPUT #3:L3$\X$='TIME USEC'
20070 R2=VAL(SEG$(L3$,21,25))\INPUT #3:L3$\INPUT #3:L3$\FOR J3=1 TO R2
20080 INPUT #3:L3$\T(J3)=VAL(SEG$(L3$,6,9))\IF T(J3)=T4 THEN 20100 \NEXT J3
20090 PRINT 'REC ';T4;'NOT FOUND IN ';D2$\GO TO 1000
20100 L5=VAL(SEG$(L3$,1,4))\H1=VAL(SEG$(L3$,21,26))\Y1$=L3$\FOR I1=L5+1 TO H1
20110 INPUT #3:L3$\NEXT I1\L6=VAL(SEG$(L3$,6,10))\N2=VAL(SEG$(L3$,27,31))
20120 CLOSE #3\Y$=SEG$(Y1$,36,47)$':'$STR$(T4)\F7=1\F8=1\F9=1\L4=(L6-1)*256+N2
20130 D3=VAL(SEG$(L3$,20,25))\IF M8=0 THEN 20150 \P8=P9+10\F7=1\F8=0\F9=0
20140 IF P8+INT((X6-X4)/8)+1<80 THEN 20150 \P8=7\Q8=Q9+5
20150 IF N2<Z5 THEN 20210 \OPEN D3$ AS FILE VF3X\X3=VAL(SEG$(L3$,12,18))
20160 X5=VAL(SEG$(L3$,33,39))\GOSUB 20650 \Q4=0\IF Q8=1 THEN 20170 \GOSUB 20250
20170 W9=0\H5=INT(N2/Z5)\X=X3\FOR I1=(L6-1)*256 TO L4-1 STEP H5\Y=VF3(I1)
20180 GOSUB 20560 \X=X+D3*H5\NEXT I1\IF Q8=1 THEN 20190 \GOSUB 20540
20190 CLOSE VF3\OPEN D3$ FOR INPUT AS FILE #3\PRINT C$'MON H'\
20200 FOR I1=1 TO R2*2+5\INPUT #3:L3$\PRINT L3$\NEXT I1\CLOSE #3\GO TO 1000
20210 PRINT 'OOD DECREASE Z5'\GO TO 1000
20220 PRINT 'NOT ENOUGH'\GO TO 20110
20230 REM
20240 REM
20250 Q9=Q8+INT(Y6/14)\P9=P8+INT(X6/8)\IF Q8<0 THEN 20270
20260 IF Q8>Q9 THEN 20270 \GO TO 20280
20270 PRINT 'ERR VRT. SIZE OR POS. Q8=';Q8;'Q9=';Q9\STOP
20280 IF P9>80 THEN 20300 \IF P8<1 THEN 20300 \IF P9<P8 THEN 20300
20290 GO TO 20310
20300 PRINT 'ERR HORZ. SIZE OR POS. P8=';P8;'P9=';P9\STOP
20310 IF F9=0 THEN 20340 \IF Q9+3<31 THEN 20330 \PRINT C$;'WOR 20'
20320 PRINT C$;'MON H;ERR WS>30';CHR$(7)\GO TO 20340
20330 PRINT C$;'WOR';Q9+3
20340 IF F8=0 THEN 20350 \PRINT C$;'ERA W'
20350 PRINT C$;'GRA ';Q8;Q9;P8;P9\P7=P8-1
20360 IF (P9-P8+1)*(Q9-Q8+1)<2200 THEN 20380
20370 PRINT CHR$(7);'GTB?'
20380 RETURN
20390 PRINT C$;'WOR H'\PRINT C$;'JUMP ';Q9+3;INT((P8+P9-LEN(X$))/2)
20400 PRINT X$;\IF P7>1 THEN 20410 \P7=2
20410 FOR I9=1 TO LEN(Y$)\PRINT C$;'JUM ';INT((Q8+Q9-LEN(Y$))/2)+I9;P7-1
20420 PRINT SEG$(Y$,I9,I9)\NEXT I9\P7=P7-2\PRINT C$;'MON H'\RETURN
20430 PRINT C$;'WOR H'\P1=P4\Q1=Q4\Q3=0\IF X7=1 THEN 20470
20440 P2=10^INT(LOG(ABS(X5-X3))/LOG(10))\P2=P2/10\P3=1\GOSUB 20760 \P2=10*P2
20450 IF P2<.5*ABS(X5-X3) THEN 20460 \P2=P2/2
20460 P3=-2\GOSUB 20760 \GO TO 20480
20470 P2=10\P3=-2\GOSUB 20760 \P3=1\FOR I7=2 TO 9\P1=I7*P4\GOSUB 20760 \NEXT I7
20480 P3=0\IF Y7=1 THEN 20520 \Q2=10^INT(LOG(ABS(Y5-Y3))/LOG(10))
20490 Q2=Q2/10\Q3=1\GOSUB 20760 \Q2=10*Q2
20500 IF Q2<.5*ABS(Y5-Y3) THEN 20510 \Q2=Q2/2
20510 Q3=-2\GOSUB 20760 \GO TO 20530
20520 Q2=10\Q3=-2\GOSUB 20760 \Q3=1\FOR I7=2 TO 9\Q1=I7*Q4\GOSUB 20760 \NEXT I7
20530 RETURN
20540 GOSUB 20740 \GOSUB 20430 \GOSUB 20390 \RETURN
20550 W9=2
20560 IF X7=1 THEN 20570 \X1=INT((X-X3)*X8+X4+.5)\GO TO 20580
20570 X1=INT(LOG(X/X3)*X8+X4+.5)
20580 IF Y7=1 THEN 20600
20590 Y1=INT((Y-Y3)*Y8+Y4+.5)\GO TO 20610
20600 Y1=INT(LOG(Y/Y3)*Y8+.5)
20610 IF W9=1 THEN 20640 \IF W9=2 THEN 20630 \W9=1
20620 X2=X1\Y2=Y1
20630 RETURN
20640 PRINT C$;'VEC';X2;Y2;X1;Y1\GO TO 20620
20650 P4=X3\Q4=Y3\IF X7=1 THEN 20660 \X8=(X6-X4)/(X5-X3)\GO TO 20690
20660 IF X5*X3<>0 THEN 20680
20670 PRINT C$;'MON H;LOG ERR L X=';X3;'R X=';X5\STOP
20680 X8=(X6-X4)/LOG(X5/X3)
20690 IF Y7=1 THEN 20700 \Y8=(Y6-Y4)/(Y5-Y3)\GO TO 20730
20700 IF Y3*Y5<>0 THEN 20720
20710 PRINT C$;'MON H;LOG ERR BOTH Y=';Y3;'TOP Y=';Y5\STOP
20720 Y8=(Y6-Y4)/LOG(Y5/Y3)
20730 RETURN
20740 X=P4\Y=Q4\GOSUB 20550 \PRINT C$;'VEC';X4;Y1;X6;Y1
20750 PRINT C$;'VEC';X1;Y4;X1;Y6\RETURN
20760 PRINT C$'WOR H'\P5=INT(((ABS(Y6-Y4)/100)+2)*P3)
20770 Q5=INT(((ABS(X6-X4)/100)+2)*Q3)\IF P5=0 THEN 20870 \Y=Q4
20780 FOR I9=0 TO 800\IF X7=1 THEN 20790 \X=P1+I9*P2\GO TO 20800
20790 X=P1*(P2^I9)
20800 GOSUB 20550 \IF X1>X6 THEN 20810 \IF X1<X4 THEN 20810 \NEXT I9
20810 FOR I8=I9-1 TO -800 STEP -1\IF X7=1 THEN 20820 \X=P1+I8*P2\GO TO 20830
20820 X=P1*(P2^I8)
20830 GOSUB 20550 \IF X1>X6 THEN 20870 \IF X1<X4 THEN 20870
20840 PRINT C$;'VEC';X1;Y1;X1;Y1+P5\PRINT C$;'VEC';X1;Y1;X1;Y1-P5
```

```
20850 IF P5>0 THEN 20860 \PRINT C$;'JUMP 'Q9+1;P8+INT((X1-12)/8)\PRINT X;
20860 NEXT I8
20870 IF Q5=0 THEN 20990 \X=P4\FOR I9=0 TO 800\IF Y7=1 THEN 20890
20880 Y=Q1+I9*Q2\GO TO 20900
20890 Y=Q1*(Q2^I9)
20900 GOSUB 20550 \IF Y1>Y6 THEN 20910 \IF Y1<Y4 THEN 20910 \NEXT I9
20910 FOR I8=I9-1 TO -800 STEP -1\IF Y7=1 THEN 20930
20920 Y=Q1+I8*Q2\GO TO 20940
20930 Y=Q1*(Q2^I8)
20940 GOSUB 20550 \IF Y1>Y6 THEN 20990 \IF Y1<Y4 THEN 20990
20950 PRINT C$;'VEC';X1;Y1;X1+Q5;Y1\PRINT C$;'VEC';X1;Y1;X1-Q5;Y1
20960 IF Q5>0 THEN 20980
20965 PRINT C$;'JUMP ';Q9-INT(Y1/14);P8-LEN(STR$(Y))\PRINT STR$(Y)
20970 IF P8-LEN(STR$(Y))-1>P7 THEN 20980 \P7=P8-LEN(STR$(Y))-1
20980 NEXT I8\PRINT C$;'MON H'
20990 RETURN
```

ST0200.BAS

The ST0200.BAS program is overlaid in response to the STORE command and serves to transfer data from the CAMAC hardware to files on the data disk. The ST0200.BAS program is discussed in Sec. 4.9.

LINES	PURPOSE
20050	set constants
20060 to 20070	enter format file title; set decimal date
20080 to 20110	reset graphics flags; input comment; check that a parameter file is in workspace
20120	use format presently in workspace
20130	save, then use format presently in workspace
20140 to 20230	load format file into workspace, allow operator to modify and save new file
20240 to 20590	read information from workspace; check and log any errors; get title; calculate disk space required for each record
20340	do not reserve disk space for 2232 ADC records
20590 to 20620	print format information into monitor; if any errors, return for modification
20630 to 20650	construct OPEN statement, store in DUMMY.BAS and overlay onto line 20890
20660 to 20670	open data header file
20680 to 20690	calculate number of lines in header file; read system time, convert to "HH:MM" form

20700 to 20840	load format information (from memory) into header file
20850 to 20880	read parameter file from workspace; strip away attributes (see 4025 User's Manual) and load into header file; close header file
20890 to 20920	open final data file and header file as virtual files (see BASIC User's Manual) and transfer information from header file to data file as virtual array elements
20930	load zeros to end of the block
20940 to 21060	prepare to read data from crate; reset 2264 buffer
21070	calculate sampling interval
21080	start 2264 readout control sequence
21090	delay to allow last readout trace to be completed
21100 to 21110	check "Q" to see if command was executed
21120 to 21150	skip to the desired first sample in buffer
21160	determine if high or low byte is to be stored
21170	transfer points from crate to data file
21200	increment V3 (points to position in data file for next data entry)
21230 to 21240	check "Q" to see if final READ was successful
21250 to 21260	execute 2264 display enable routine
21300 to 21310	update INITIA.LIZ file (increment shot number), return to OVRNEW.BAS for next command

21320 to 21350 subroutine to check if a format file is in the workspace before trying to use it

```
02-JAN-80 COPY OF ST0200.BAS, OPTION I    LPRINT V1:BB

20050 F6=101\D1$=' 00,  1, '\L5=1001
20060 PRINT B2$'FORMAT FILE ';\INPUT F1$
20070 F1$='PGM:'$F1$'.FMT'\D2$=D4$$SEG$(STR$((S5+1)/1000+1)$'000',2,5)
20080 Q8=0\M8=0\P8=7\Q8=3\PRINT 'FILE COMMENT'\INPUT C0$\GOSUB 9400
20090 IF F3$=F2$ THEN 20120 \PRINT C$'WOR 20 H'\OPEN P1$ AS FILE #2\L2=0
20100 INPUT #2:L3$\PRINT L3$\L2=L2+1\IF END #2 THEN 20110 \GO TO 20100
20110 S2=L2+2\CLOSE #2\PRINT C$'MON H'\GOSUB 9400 \F2$=F3$\PRINT C$'WOR H'
20120 IF F1$<>'PGM:W.FMT' THEN 20130 \GOSUB 21320 \GO TO 20240
20130 IF F1$='PGM:WN.FMT' THEN 20190 \PRINT C$'WOR H'C$'JUM';S2
20140 PRINT C$'DLI 100'\PRINT 'FORMAT FILE ';F1$\OPEN F1$ AS FILE #1
20150 INPUT #1:L3$\PRINT L3$\IF END #1 THEN 20160 \GO TO 20150
20160 CLOSE #1\PRINT "***EAT AT JOE'S***"
20170 PRINT C$'MON H'\PRINT 'MAKE CHANGES NOW'\PRINT 'SAVE NEW FORMAT'\INPUT L3$
20180 IF SEG$(L3$,1,1)<>'Y' THEN 20240
20190 GOSUB 21320 \PRINT 'FORMAT FILE NAME'
20200 INPUT F1$\F1$='PGM:'$F1$'.FMT'\OPEN F1$ FOR OUTPUT(3) AS FILE #1
20210 S1=S2+1\GOSUB 9200 \INPUT L3$\PRINT #1:L3$\INPUT L3$\PRINT #1:L3$
20220 R2=VAL(SEG$(L3$,8,13))\INPUT L3$\PRINT #1:L3$
20230 FOR I4=1 TO R2\INPUT L3$\PRINT #1:L3$\NEXT I4\GOSUB 9300 \CLOSE #1
20240 S1=S2+2\GOSUB 9200 \INPUT F1,R2,D3\INPUT L3$
20250 E2=0\E3=0\OPEN 'PGM:TITLE.REC' AS FILE VF1$=16
20260 L6(1)=INT((R2*94+275+L2*48)/512+2)\FOR J3=1 TO R2\S7=0
20270 INPUT T(J3),U1(J3),C3(J3),G1(J3),O1(J3),N3,N4,F1(J3),N2(J3),E1(J3),S1(J3)
20280 C1(J3)=N3\I1(J3)=N4\L1$(J3)='TITLE NOT FND IN "TITLE.REC"'
20290 IF T(J3)>999 THEN 20300 \IF T(J3)<0 THEN 20300 \GO TO 20310
20300 E2=1\PRINT 'ERR. REC#';J3;' UNIT';U1(J3);':  ILLEGAL TYPE'\PRINT
20310 IF Q4(U1(J3))=2232.1 THEN 20510 \IF Q4(U1(J3))=2264 THEN 20350
20320 REM EXPAND HERE FOR NEW EQUIPMENT
20330 E2=1\PRINT 'ERR. REC#';J3;': UNIT';U1(J3);' NOT FND IN "CONFIG.INF"'\PRINT
20340 L6(J3)=0\L6(J3+1)=L6(J3-1)\N2(J3)=0\GO TO 20560
20350 S8=Q3(U1(J3))\GOSUB 2912 \F=1\GOSUB 7900 \N=D\GOSUB 2500 \N6(J3)=FNC(X)
20360 IF N6(J3)<>1 THEN 20370 \E2=1\PRINT 'ERR. UNIT';U1(J3);': NO 1 CHNNL DATA'
20370 PRINT \IF C3(J3)<=N6(J3) THEN 20390 \E2=1
20380 PRINT 'ERR. REC#';J3;' UNIT';U1(J3);':  CHNL#';C3(J3);' NOT SAMPLED'\PRINT
20390 S8=Q3(U1(J3))\S7=0\GOSUB 2912 \F=1\GOSUB 7900 \N=D\GOSUB 2500
20400 IF B(10)=0 THEN 20410 \E2=1\PRINT 'ERR. UNIT';U1(J3);': RATE TOO HI'\PRINT
20410 R4(J3)=FNT(X)\M2(J3)=FNP(X)*Q5(U1(J3))*1024
20420 F1(J3)=INT(F1(J3)/R4(J3))*R4(J3)
20430 IF M2(J3)<=(2^15/N6(J3)+F1(J3)/R4(J3)) THEN 20450 \E2=1
20440 PRINT 'ERR. REC#';J3;' UNIT';U1(J3);':  ILLEGAL FIRST TIME'\PRINT
20450 IF E1(J3)=0 THEN 20490 \A3(J3)=INT(2^15/N6(J3)-M2(J3)+F1(J3)/R4(J3))
20460 E1(J3)=INT(E1(J3)/R4(J3)*R4(J3)\IF INT(N2(J3)/256)=N2(J3)/256 THEN 20480
20470 N2(J3)=INT((N2(J3)+256)/256)*256
20480 S1(J3)=(E1(J3)-F1(J3))/N2(J3)\GO TO 20500
20490 S1(J3)=INT(S1(J3)/R4(J3))*R4(J3)\E1(J3)=F1(J3)+(N2(J3)-1)*S1(J3)
20500 L6(J3+1)=L6(J3)+N2(J3)/256\GO TO 20560
20510 L6(J3)=0\L6(J3+1)=L6(J3-1)
20520 IF C3(J3)<0 THEN 20530 \IF C3(J3)>31 THEN 20530 \GO TO 20540
20530 E2=1\PRINT 'ERR. REC#';J3;' UNIT';U1(J3);':  ILLEGAL CHNNL'\PRINT
20540 F=INT(C3(J3)/16)\S7=C(J3)-F*16\GOSUB 2912 \GOSUB 7900 \S1(J3)=D/4096*10-5
20550 F1(J3)=550+C3(J3)*550\GO TO 20560
20560 FOR I1=1 TO 999\L3$=VF1(I1)\IF VAL(SEG$(L3$,1,3))=T(J3) THEN 20580
20570 IF SEG$(L3$,5,LEN(L3$))='END' THEN 20590 \NEXT I1
20580 L1$(J3)=SEG$(L3$,5,16)
20590 NEXT J3\PRINT 'REC#','UNIT#','1ST TIME','LAST TIME','STEP TIME','# SAMPLS'
20600 CLOSE VF1\FOR J3=1 TO R2\PRINT J3,U1(J3),F1(J3),E1(J3),S1(J3),N2(J3)
20610 NEXT J3\GOSUB 9300 \IF E2=0 THEN 20620 \GO TO 20170
20620 L4=L6(R2+1)-1\H1=J3+1006\PRINT 'FORMAT# ';F1;' #REC=';R2;'  SHOT# ';S5+1
20630 OPEN 'PGM:DUMMY.BAS' FOR OUTPUT(1) AS FILE #2
20640 PRINT #2:'20890 OPEN "';D2$;'" FOR OUTPUT(';STR$(L4);') AS FILE VF2%'
20650 CLOSE #2\PRINT 'START OVERLAY'\OVERLAY "PGM:DUMMY.BAS"
20660 PRINT 'FINISH'\OPEN 'PGM:DATA.HED' FOR OUTPUT(10) AS FILE #2
20670 PRINT 'DATA FILE LENGTH = ';L4;' BLOCKS'
20680 L3=L2+R2*2+5\GOSUB 9500
20690 T2$=STR$(INT(T1/3600))$':'$STR$(INT((T1-INT(T1/3600)*3600)/60))
20700 PRINT #2:FNQ(L5,3);' F# FFL#  SHT#  #REC  CNT  CPY    DATE   REC.LST';
20710 PRINT #2:'  PARM.LST   TIME   #LINES'\L5=L5+1
20720 PRINT #2:FNQ(L5,3)FNQ(F6,2)FNQ(F1,3)FNQ(S5+1,4)FNQ(R2,4)D1$;' 'D4$;',';
20730 PRINT #2:FNQ(1005,6);FNQ(2*R2+1008,7)FNQ(T1,8)FNQ(L3,4)\L5=L5+1
20740 PRINT #2:FNQ(L5,3);'TIME 'T2$', 'N1$', 'C0$\L5=L5+1
20750 C8=2
20760 PRINT #2:FNQ(L5,3);'TYPE  CAT  ID#  HEADR   LEN    TITLE'\L5=L5+1
20770 FOR J3=1 TO R2
20780 C0$=' BLK#  1ST-TM   STEP #SMPL  LST-TM   GAIN OFSET ERFLG CKSUM SRIAL#'
20790 PRINT #2:FNQ(L5,3)FNQ(T(J3),3)FNQ(C1(J3),3)FNQ(I1(J3),3)FNQ(H1,5)FNQ(1,5);
20800 PRINT #2:' 'L1$\L5=L5+1\H1=H1+1\NEXT J3\PRINT #2:FNQ(L5,3);C0$
20810 L5=L5+1\FOR J3=1 TO R2
20820 PRINT #2:FNQ(L5,3)FNQ(L6(J3),4)FNQ(F1(J3),6)FNQ(S1(J3),5)FNQ(N2(J3),4);
```

```
20830 PRINT #2:FNQ(E1(J3),6)FNQ(G1(J3),6)FNQ(O1(J3),4)FNQ(E2,3)FNQ(S3,5);
20840 PRINT #2:FNQ(Q6(U1(J3)),5)\L5=L5+1\NEXT J3\S1=1\GOSUB 9200 \FOR I5=1 TO L2
20850 INPUT L3$
20860 S1=POS(L3$,C$,1)\IF S1=0 THEN 20880
20870 L3$=SEG$(L3$,1,S1-1)&SEG$(L3$,POS(L3$,';',1)+1,LEN(L3$))\GO TO 20860
20880 PRINT #2:FNQ(L5,3);L3$\L5=L5+1\NEXT I5\GOSUB 9300 \PRINT #2;'EEE'
20890 OPEN "SHOT6" FOR OUTPUT(5) AS FILE VF2%\REM THIS LINE UPDATED EACH RUN
20900 CLOSE #2\OPEN 'PGM:DATA.HED' FOR INPUT AS FILE VF1%
20910 FOR I1=0 TO (L6(1)-1)*256-1\IF VF1(I1)=17733 THEN 20930 \VF2(I1)=VF1(I1)
20920 NEXT I1
20930 CLOSE VF1\FOR I2=I1 TO (L6(1)-1)*256-1\VF2(I2)=0\NEXT I2
20940 V3=(L6(1)-1)*256-1\FOR J3=1 TO R2\IF U1(J3)=3 THEN 21300
20950 C8=2\S8=Q3(U1(J3))\S7=0\GOSUB 2912 \GO TO 21060
20960 REM
20965 REM
20970 REM
20980 REM
20990 REM
21000 REM
21010 REM
21020 REM
21030 REM
21040 REM
21050 REM
21060 GOSUB 5000
21070 H5=INT(S1(J3)/R4(J3))
21080 F=26\GOSUB 7900 \S7=FNA(C3(J3))\GOSUB 2912
21090 N3=SIN(1)+COS(1)+SIN(2)
21100 F=8\GOSUB 7900 \CALL 'PEEK'(A8,N)\GOSUB 2500 \IF B(8)=1 THEN 21120
21110 PRINT 'NO RESPONSE FROM(';C8;S8;')'
21120 F=10\GOSUB 7900 \F=16\GOSUB 7900
21130 CALL 'FAST'(A9,0,A3(J3),0,Q,D)
21140 PRINT 'REC# '#J3,'FROM '#A3,'STEP '#H5,N2(J3);' SAMPLES'
21150 REM
21160 U2=INT(1-2*(C3(J3)-2*INT(C3(J3)/2)))
21170 FOR I1=V3+1 TO N2(J3)+V3\CALL 'FAST'(A9,U2,H5,256,Q,Y)\VF2(I1)=Y\NEXT I1
21180 REM
21190 REM
21200 V3=V3+N2(J3)
21210 REM
21220 REM
21230 IF Q=1 THEN 21250
21240 PRINT 'OOD IN ('#Q3(U1(J3))#C3(J3)#') REC#'#J3
21250 GOSUB 5000
21260 F=10\GOSUB 7900 \F=24\GOSUB 7900 \F=16\GOSUB 7900
21270 GO TO 21300
21280 REM
21290 REM
21300 NEXT J3\CLOSE VF2\OPEN 'INITIA.LIZ' AS FILE VF2%=16\S5=S5+1\L3$=STR$(S5)
21310 VF2(2)=L3$\PRINT 'DATA FILE '#D2$#' = '#L4#' BLOCKS'\CLOSE VF2\GO TO 1000
21320 F3$=''\FOR I5=1 TO 6\PRINT C$'JUM'S2;I5\PRINT C$'REP 01'\PRINT C$'UP 1'
21330 INPUT L3$\F3$=F3$&SEG$(L3$,17,17)\NEXT I5\IF F3$='FORMAT' THEN 21350
21340 PRINT 'NO FORMAT IN WORK SPACE'\GO TO 20060
21350 RETURN
```

```
02-JAN-80 COPY OF GRA200.BAS, OPTION I    LPRINT V1:BB

20050 GOSUB 5000 \PRINT 'CHANNEL'\INPUT C3
20060 Y$='CHANNEL '$STR$(C3)\F7=1\F8=1\F9=1\IF M8=0 THEN 20090
20070 P8=P9+10\F7=1\F8=0\F9=0\IF P8+INT((X6-X4)/8)+1<80 THEN 20090
20080 P9=7\Q8=Q9+5
20090 F=1\GOSUB 7900
20100 N=D\GOSUB 2500 \N8=FNC(X)\IF C3>N8 THEN 20210
20110 PRINT '# SMPLS? (MULTIPLES OF'$Z5$'), 1ST SMPL #?'\INPUT H4,A3\A3=INT(A3)
20120 IF H4<Z5 THEN 20220 \F=26\GOSUB 7900 \F=10\GOSUB 7900
20130 S7=FNA(C3)\GOSUB 2912 \F=16\GOSUB 7900 \CALL "FAST"(A9,0,A3,0,Q,D)
20140 U2=INT(1-2*(C3-2*INT(C3/2)))\X3=A3\X5=A3+H4\GOSUB 20650 \Q4=0
20150 IF Q8=1 THEN 20160 \GOSUB 20250
20160 W9=0\H5=INT(H4/Z5)\X=A3\FOR I1=0 TO Z5\CALL 'FAST'(A9,U2,H5,256,Q,Y)
20170 GOSUB 20560 \X=X+H5\NEXT I1\IF Q8=1 THEN 20190 \GOSUB 20540
20190 GOSUB 5000 \S7=0\F=24\GOSUB 7900 \S7=FNA(C3)\F=16\GOSUB 7900
20200 GO TO 1000
20210 PRINT 'CHNNL NOT SMPLD'\GO TO 1000
20220 PRINT 'NOT ENOUGH'\GO TO 20110
20230 REM
20240 REM
```

```
02-JAN-80 COPY OF FIXRED.BAS, OPTION I    LPRINT V1:BB

20050 PRINT B2$\PRINT 'DATA FILE'\INPUT D3$\PRINT 'RECORD TYPE'\INPUT T4
20060 OPEN D3$ FOR INPUT AS FILE #3\INPUT #3:L3$\INPUT #3:L3$\X$='TIME USEC'
20070 R2=VAL(SEG$(L3$,21,25))\INPUT #3:L3$\INPUT #3:L3$\FOR J3=1 TO R2
20080 INPUT #3:L3$\T(J3)=VAL(SEG$(L3$,6,9))\IF T(J3)=T4 THEN 20100 \NEXT J3
20090 PRINT 'REC '$T4$'NOT FOUND IN '$D2$\GO TO 1000
20100 L5=VAL(SEG$(L3$,1,4))\H1=VAL(SEG$(L3$,21,26))\Y1$=L3$\FOR I1=L5+1 TO H1
20110 INPUT #3:L3$\NEXT I1\L6=VAL(SEG$(L3$,6,10))\N2=VAL(SEG$(L3$,27,31))
20120 CLOSE #3\Y$=SEG$(Y1$,36,47)&':'$STR$(T4)\F7=1\F8=1\F9=1\L4=(L6-1)*256+N2
20130 D3=VAL(SEG$(L3$,20,25))\IF M8=0 THEN 20150 \P8=P9+10\F7=1\F8=0\F9=0
20140 IF P8+INT((X6-X4)/8)+1<80 THEN 20150 \P8=7\Q8=Q9+5
20150 IF N2<Z5 THEN 20210 \OPEN D3$ AS FILE VF3%\X3=VAL(SEG$(L3$,12,18))
20160 X5=VAL(SEG$(L3$,33,39))\GOSUB 20650 \Q4=0\IF Q8=1 THEN 20170 \GOSUB 20250
20170 W9=0\H5=INT(N2/Z5)\X=X3\FOR I1=(L6-1)*256 TO L4-1 STEP H5\Y=VF3(I1)
20180 GOSUB 20560 \X=X+D3*H5\NEXT I1\IF Q8=1 THEN 20190 \GOSUB 20540
20190 CLOSE VF3\OPEN D3$ FOR INPUT AS FILE #3\PRINT C$'MON H'\
20200 FOR I1=1 TO R2*2+5\INPUT #3:L3$\PRINT L3$\NEXT I1\CLOSE #3\GO TO 1000
20210 PRINT 'OOD DECREASE Z5'\GO TO 1000
20220 PRINT 'NOT ENOUGH'\GO TO 20110
20230 REM
```

GRA200.BAS and FIXRED.BAS are identical to the first 19 lines of PLT200.BAS and RED200.BAS, respectively, but allow for faster overlaying if the graphics routines are already resident.

TABLE A-2

Subroutines Used in Storage Routines

These subroutines reside in OVRNEW.BAS and are available to all overlays.

Address	Function
2000	separate a 16 bit data word into upper and lower bytes (normally used with PEEK)
2500	load the 16 bit CSR word from the crate controller into the array B(1) to B(16)
2912	calculate the total CAMAC address from the crate, station, and subaddress numbers
5000	reset 2264 digitizer logic by executing 17000 "reads"
7900	execute CAMAC F commands
9200	set 4025 in buffered mode and prepare the workspace to be read by the computer starting at line S1
9300	reset 4025 after computer has read from workspace
9400	load F3$ with the first 20 characters in the workspace
9500	set T1 equal to the system time (seconds after midnight)

CREATE.BAS

The CREATE.BAS program is not part of the storage routine overlays and must be run on its own. This program creates the necessary files on a blank initialized data disk. These files are discussed in Sec. 4.4. The CREATE.BAS program prints instructions to the operator during executions and is discussed in Sec. 5.4.

LINES	PURPOSE
5	check system date, abort if not set
6	enter disk sequence number; end program if less than or equal to zero
9 to 40	create a 4 block CATALO.G file; fill with "@" characters using virtual string arrays - This step will create a vacant space of 3 blocks filled with a consistent pattern of data (for neatness). When the real CATALO.G file is created in step 61, this space of 4 blocks is left vacant. Next, when the TITLE.TXT file is created, RT-11 will place it at the beginning of this empty space, leaving 3 free blocks. These 3 blocks guarantee that the files created above can be modified without being repositioned at higher block numbers on the disk (e.g. will not be mixed with data files).

50 to 60	create a 1 block INITIA.LIZ file; enter the message; "EMPTY DATA DISK"
61 to 65	create the real CATALO.G file (2 blocks long); enter the message "DATA ON THIS DISK HAVE NOT BEEN CATALOGED"
80	assemble disk title DAXXX where XXX is the disk sequence number entered by operator
90 to 162	create a 1 block TITLE.TXT file containing the disk title, sequence number, and comment entered by operator (up to 450 characters)
165	check length of comment
170 to 400	return for next disk (enter zero to exit program)

```
02-JAN-80 COPY OF CREATE.BAS, OPTION I    LPRINT V1:BB

5 IF DAT$<>'          ' THEN 6 \PRINT 'SET THE DATE YOU DUMMY. GO TO RT-11'\CRASH
6 PRINT 'ENTER DISC #(JUST THE NUMBER)'$\INPUT N\IF N>0 THEN 9 \GO TO 300
9 PRINT 'PLEASE WAIT 25 SECONDS'
10 OPEN 'CATALO.G' FOR OUTPUT(4) AS FILE VF1$=64
20 FOR I=0 TO 31
30 VF1(I)='@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@'
40 NEXT I
50 CLOSE \OPEN 'INITIA.LIZ' FOR OUTPUT(1) AS FILE #1
60 PRINT #1:'                    EMPTY DATA DISC                    '\CLOSE #1
61 OPEN 'CATALO.G' FOR OUTPUT(2) AS FILE #1
62 PRINT #1:'DATA ON THIS DISK HAS NOT BEEN CATALOGED'
63 FOR I=1 TO 12\PRINT #1:'
64 NEXT I
65 CLOSE #1
80 T1$='000'$STR$(N)\T1$='DA'$SEG$(T1$,LEN(T1$)-2,LEN(T1$))
90 PRINT 'ENTER COMMENTS FOR '$T1$;' ENDING WITH "END" AS THE FIRST THREE'
100 PRINT ' CHARACTERS OF A NEW LINE'
110 OPEN 'TITLE.TXT' FOR OUTPUT(1) AS FILE #1
120 L=0
130 INPUT L$\IF L$='' THEN 131 \IF L$='END' THEN 131 \GO TO 140
131 PRINT 'PLEASE ENTER COMMENT'\GO TO 130
140 PRINT #1:T1$$' --INITIALIZED '$DAT$$'--'$SEG$(L$,1,48)\L=31+LEN(L$)+2
145 IF LEN(L$)<=48 THEN 160
150 PRINT #1:SEG$(L$,49,LEN(L$))\L=L+2
160 INPUT L$\IF L$='END' THEN 170 \IF L+LEN(L$)>509 THEN 165
162 PRINT #1:L$\L=L+LEN(L$)+2\GO TO 160
165 PRINT ' TOO MUCH. PLEASE RETYPE LAST LINE WITH '$509-L;' OR FEWER CHARS.'
166 GO TO 160
170 CLOSE #1\PRINT 'INITIALIZATION OF 'T1$' IS COMPLETE.'
180 PRINT ' ENTER 0 FOR NEXT DISC # TO EXIT. '$\GO TO 6
300 CLOSE \PRINT 'FINISHED'\STOP
400 END
```

CATDIR.BAS

The CATDIR.BAS program along with the CATDIR.COM command file are used at the end of a data run to enter information into the CATALO.G file on each data disk. This process is discussed in Sec. 4.4.

The CATDIR.COM command file is executed by typing "@CATDIR". This reads the information in the directory on the data disk and stores it in a temporary file, DIRECT.TMP, on the system disk. The argument of this command DS1:79%%%.* needs to be updated each year e.g. DX1:80%%%.* .

The CATDIR.BAS program takes the information from the temporary file and transfers it to the CATALO.G file on the data disk.

LINES	PURPOSE
10 to 20	set characters C1$ and B$; calculate number of entries that will fit in CATALO.G
100 to 110	open the temporary directory file on system disk
120 to 190	check to see if data disk has already been cataloged, if so inform operator
200 to 210	move CATALO.G file out of reserved position
220 to 240	check length of temporary file; abort if too long for CATALO.G
300 to 340	transfer information from the temporary file to the CATALO.G file and also to the terminal

350 to 370	print error message if number of files in CATALO.G is different from that in the temporary file
400	end of program
1000	function to columnize printout, right justified

```
02-JAN-80 COPY OF DX1:CATDIR.BAS, OPTION ILPRINT V1:BB

10 C1$='DX1:CATALO.G'\B$=CHR$(7)$'***WARNING-'
20 B=2\M=INT(B*512/21)+2\REM: B=#BLOCKS IN CATALO.G **SEE ALSO 'OPEN' LINE 220
100 OPEN 'DX0:DIRECT.TMP' FOR INPUT AS FILE #1
110 OPEN C1$ FOR INPUT AS FILE #2
120 INPUT #2:L$
130 IF SEG$(L$,1,4)='DATA' THEN 200
140 PRINT B$'THIS DISK MAY HAVE BEEN CATALOGUED BEFORE'
150 PRINT 'THE FIRST LINE OF '$C1$;' IS---'
160 PRINT L$
170 PRINT 'DO YOU STILL WANT TO CATALOGUE IT (Y/N) '$
180 INPUT A$\IF SEG$(A$,1,1)='Y' THEN 200
190 STOP
200 CLOSE #2\OPEN C1$ FOR OUTPUT(1) AS FILE #2
210 PRINT #2:'DUMMY'\CLOSE #2\REM: THIS MOVES FILE OUT OF RESERVED POSITION
220 OPEN C1$ FOR OUTPUT(2) AS FILE #2
230 FOR I=1 TO M\IF END #1 THEN 300 \INPUT #1:L$\NEXT I
240 PRINT B$;'TOO MANY DATA FILES FOR'$B;'BLOCKS. MAX IS '$M-4\STOP
300 N=I-4\RESTORE #1\INPUT #1:L$\REM:DISCARD DATE
304 PRINT
305 FOR U=0 TO 2 STEP 2
310 PRINT #U:FNJ(N,3)$' FILES '$DAT$
312 PRINT #U:'FILENAME  LEN START'
315 REM:       XXXXXX.FFF,123,1234
317 NEXT U
320 FOR I=1 TO N\INPUT #1:L$\FOR U=0 TO 2 STEP 2
330 PRINT #U:SEG$(L$,1,10)$','$FNJ(VAL(SEG$(L$,14,16)),14)$','$
340 PRINT #U:FNJ(VAL(SEG$(L$,29,32)),19)\NEXT U\NEXT I
345 PRINT
350 INPUT #1:L$\IF VAL(SEG$(L$,1,POS(L$,' ',3)))=N THEN 370
360 PRINT B$;
370 PRINT 'NO. OF FILES FOUND= '$STR$(N)$'. DIRECTORY SAID '$L$
400 CLOSE \PRINT 'CATALOGUING COMPLETE'\STOP
1000 DEF FNJ(X,C)=TAB(C-LEN(STR$(X)))$STR$(X)\REM: PRINTS X ENDING IN COLUMN C
```

```
02-JAN-80 COPY OF DX1:CATDIR.COM, OPTION ILPRINT V1:BB

DIR/COL:1/BLOCKS/ALPHABET/OUTPUT:DX0:DIRECT.TMP DX1:79XXXX.*
```

APPENDIX B

USER FUNCTION ADDITIONS

TO BASIC

The following RT-11 assembly language (MACRO) subroutines have been written to add new capabilities to the DEC supplied BASIC. The following listings are for BASIC V01 B (old BASIC). Because the "POKE", "STOB", "FAST", and especially "EXEC" can change memory contents, they should be used with caution. A large amount of error checking is performed, making the subroutines appear more complicated than they are. Error recovery is attempted wherever possible, except in "FAST". It is inconsistent with the purpose of "FAST" to spend a lot of time correcting errors, so the routine simply crashes back to BASIC, giving an error message. The most likely source of error is that the arguments were in floating point format, not integer format.

"EXEC" is a powerful tool for developing new subroutines without going through the lengthy process of assembling and linking BASIC. Machine code is entered directly into vacant memory locations (see "FREM") and execution commences at the address given in the call to "EXEC". One argument can be passed to the machine language code via the call to "EXEC"; it is placed in register R3 when execution commences. Other arguments can be passed (both ways) by "PEEK" and "POKE". The code must end with "RTS PC" = 207 (octal) or 135 (decimal).

Because this method of calling subroutines is so general, it is subject to operator error and is not recommended for inexperienced users or for permanent systems.

The assembly and linking procedures followed are given in Appendix F of the BASIC Language Reference Manual. The subroutine set "GETARG" in Appendix D of the BASIC manual is used in these routines. The table of subroutine names and addresses used, "SUBTAB.MAC", and the RT-11 commands used to link the routines are reproduced after the subroutine listings.

```
02-JAN-80 COPY OF DX1:MEMOPS.MAC, OPTION ILPRINT V1:BB

;MEMOPS      PEEK AND POKE ,GETB STOB,EXEC USER FUNCTIONS FOR BASIC V2
        .TITLE MEMOPS
        .GLOBL PEEKST,POKEST,GETBST,STOBST,EXECST
        .GLOBL GETARG,STORE,INT,MSG
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
FAC1=40
FAC2=42
;   PEEK (A,C)    A IS THE ADDRESS,(SUPPLIED),C IS THE CONTENTS
;RETURNED
PEEKST: MOV     #TABLEP,R0
        JSR     PC,GETARG
        .BYTE   1,2,0           ;THE ZERO TERMINATES ARG LIST
        .EVEN                   ;MAKE SURE NEXT ADDRESS EVEN
        MOV     #FAC1,R3
        ADD     R5,R3           ;R3 POINTS TO FAC1 (MSB)
        MOV     #A1,R2          ;R2 POINTS TO MSB A
        JSR     PC,FPTEST               ;NO IN RANGE?
        JSR     PC,WORTST               ;IS IT A WORD ADDRESS?
        CLR     (R3)+           ;CLEAR MSB FAC,R3 POINTS TO FAC2
        MOV     (R2),(R3)               ;STORE CONTENTS IN FAC2
        MOV     #C,R0           ;R0 POINTS TO VARIABLE INFO IN TABLEP
        JSR     PC,STORE
        RTS     PC
;
;
POKEST: MOV     #POKTAB,R0
        JSR     PC,GETARG
        .BYTE   1,1,0           ;TWO INPUTS,ZERO TERMINATES LIST
        .EVEN
        MOV     #P2,R2          ;GET DAT POINTER
        JSR     PC,FPTEST               ;TEST & GET DATA TO R3
        MOV     R2,R3
        MOV#P1,R2               ;GET ADDRESS POINTER
        JSR     PC,FPTEST               ;TEST FOR FP FORMAT
        JSR     PC,WORTST
        MOV     R3,(R2)         ;STORE WORD
        RTS     PC
;
;
;
;
;
        WORTST
;SUBROUTINE VIA PC ROUNDS DOWN ODD VALUES OF R2 AND PRINTS WARNING
;
;
;
WORTST: BIT     R2,#1           ;ODD ADDRESS  MAKES  Z CLEAR
        BEQ     EVENAD
        JSR     R1,MSG          ;WARNING
        .ASCIZ  'ODD ADDRESS ROUNDED DOWN'
;BUT IT CONTINUES ANYWAY
        .EVEN
        BIC     #1,R2           ;MAKE ADDRESS EVEN
EVENAD: RTS     PC              ;RETURN WITH EVEN ADDR IN R2
;
;
;
;
                        GETB            88888888888888888888**********
GETBST: MOV     #TABLEP,R0
        JSR     PC,GETARG               ;TWO ARGS
        .BYTE   1,2,0           ;1IN 1OUT
        .EVEN
        MOV     #A1,R2          ;GET ADDR
        JSR     PC,FPTEST               ;BOMB IF TOO BIG
        MOV     #FAC1,R3
        ADD     R5,R3           ;R3 POINTS TO FAC1
        CLR     (R3)+           ;UPPER FAC CLEARED
        CLR     (R3)            ;UPPER BYTE LOWER FAC CLR
        MOVB    (R2),(R3)
        MOV     #C,R0           ;RETURN ARGUMENT
```

```
        JSR     PC,STORE                ;TO C
        RTS     PC
;
;
;
;*****************************************************************
;
;
;
;SUBROUTINE TO INTEGERIZE FP ARG POINTED TO BY R2,RETURNS IN R2 .
;
;USES FAC,.BOMBS IF NO OUTSIDE 32767,-32768
FPTEST: MOV     R3,-(SP)                ;SAVE R3
        MOV     #FAC1,R3
        ADD     R5,R3
        MOV     (R2)+,(R3)+
        MOV     (R2),(R3)               ;ARG @ R2 GOT TO FAC
        JSR     PC,INT                  ;INTEGERIZE
        TST     -(R3)           ;TEST FOR INTEGER FOMAT
        BEQ     INTFMT
        TRAP    0                       ;BOMBS IF FP FMT
        .ASCIZ  'WORD >32767 OR < -32768(DECIMAL)'
        .EVEN
INTFMT: TST     (R3)+           ;GO BACK TO FAC2
        MOV     (R3),R2         ;ADDRESS  IS IN R2
        MOV     (SP)+,R3                ;RESTORE OLD R3
        RTS     PC              ;RETURN
;
;
;
:
STORST: MOV     #POKTAB,R0
        JSR     PC,GETARG
        .BYTE   1,1,0
        .EVEN
        MOV     #P2,R2
        JSR     PC,FPTEST
        MOV     R2,R3
        BIT     #177400,R3              ;IS UPPER BYTE CLEAR?
        BEQ     BYTEFM          ;IF NOT PROCEED
        MOV     R3,-(SP)                ;SAVE R3
        JSR     R1,MSG          ;IF SO GIVE WARNING
        .ASCIZ  'UPPER BITS CLEARED'
        .EVEN
        MOV     (SP)+,R3                ;RESTORE R3
BYTEFM: BIC     #177400,R3
        MOV     #P1,R2          ;GET ADDRESS
        JSR     PC,FPTEST               ;TO R2
        MOVB    R3,(R2)         ;STORE BYTE
        RTS     PC
;
;
;
EXECST: MOV     #POKTAB,R0              ;'EXEC'(ADDR,INPUT)
        JSR     PC,GETARG               ;CAUSES AN EFFECTIVE JSR PC TO
        .BYTE   1,1,0           ;ADDRESS
        .EVEN
        MOV     #P2,R2
        JSR     PC,FPTEST
        MOV     R2,R3           ;INPUT IN R3
        MOV     #P1,R2
        JSR     PC,FPTEST
        JMP     (R2)                    ;THE SUBROUTINE RETURN CODE(207)=135DECIMAL
                                        ;IS PROVIDED BY THE USER IN THE STRING OF CODE
;
;
;
; ARGUMENT AREA FOR PEEK
TABLEP:
A1:     .WORD   0,0             ;VALUE OF A
C:      .WORD   0,0,0           ;ADDRESS INFO FOR C.3 WORDS
POKTAB:
P1:     .WORD   0,0             ;VALUE OF ADDRESS
P2:     .WORD   0,0             ;DATA
;
.=TABLEP
;
;
;
        .END
```

```
02-JAN-80 COPY OF DX1:MEMOP2.MAC, OPTION ILPRINT V1:BB

;       FAST & FREM.....BASIC SUBROUTINES FOR FAST READING OF
;               THE 2264 DIGITIZER AND FOR INSERTION OF MACHINE CODE
;               NOTE: THE MEMORY SET ASIDE FOR MACHINE CODE IS IN A SEPARATE
;       MACRO TO BE LINKED INTO THE ROOT SEGMENT IF AN OVERLAY STRUCTURE IS USED
;               NOTE: FASTTB IS ONLY DECLARED A GLOBAL SO THAT IS GIVEN ON THE
;                 LOAD MAP
;
;
                .TITLE MEMOP2
                .GLOBL  FREEST,FASTST,INTSTO
                .GLOBL  GETARG,STORE,FIRSTA,LASTA,FASTTB
;
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
FAC1=40
FAC2=42
;     FREM  CALL'FREM'(FIRSTADDR,LASTADDR) RETURNS THE FIRST AND
;       LAST AVAILABLE ADDRESSES IN A RESERVED BLOCK
FREEST: MOV     #TABFRE,R0          ;'FREM'(FIRSTA,LASTA)
        JSR     PC,GETARG           ;
        .BYTE   2,2,0               ;
        .EVEN                       ;
        MOV     #FIRSTA,R0        ; RETURN FIRST ADDRESS
        JSR     R4,INTSTO
        .WORD   F1
        MOV     #LASTA,R0           ; RETURN LAST ADDRESS
        JSR     R4,INTSTO
        .WORD   F2
        RTS     PC

TABFRE:
F1:             .WORD 0,0,0         ; ROOM FOR FIRSTA
F2:             .WORD 0,0,0         ; ROOM FOR LASTA
;THE FOLLOWING LINES HAVE BEEN REMOVED TO A SEPARATE MACRO
;FIRSTA:          .BLKW 27           ; AREA FREE FOR USER
;LASTA:           .WORD 0            ; LAST WORD FREE
;                                    ; TOTAL OF 24(10) WORDS
;
;
;       FAST... CALL 'FAST'(ADDR,HILO,STEP,ERVAL,Q,DATA)
;       DOES A READ AT ADDR VIA AN F2,THEN ADVANCES SO THAT A TOTAL
;       OF 'STEP' READS ARE DONE. Q IS RETURNED FOR THE READ OP, NOT
;       THE ADVANCES. IF NO Q, ERVAL IS RETURNED IN PLACE OF DATA
;       HILO=0 GIVES FULL WORD
;       HILO=1 GIVES UPPER BYTE
;       HILO=-1 GIVES LOWER BYTE
;       STEP=0 IS LEGAL, BUT NO DATA IS RETURNED(Q,DATA ARE UNCHANGED)
FASTST: MOV     #FASTTB,R0
        JSR     PC,GETARG
        .BYTE   1,1,1,1,2,2,0
        .EVEN
        MOV     @#ADDH,R0           ; CHECK ALL UPPER WORDS
        BIS     @#HILOH,R0          ; TO DETECT FLOATING POINT
        BIS     @#ERVALH,R0         ; USE LOGICAL 'OR' TO SAVE CODE
        BIS     @#STEPH,R0          ; NO CORRECTIVE ACTION TAKEN BECAUSE THIS

        BNE     FPERMS              ; IS SUPPOSED TO BE FAST--- JUST BOMB OUT
;
        TST     @#STEPL             ; TEST FOR ZERO STEP
        BEQ     RETURN              ; NO OP IF STEP=0
;
        MOV     @#ADDL,R2           ; R2 IS CAMAC MODULE ADDRESS
        MOV     R2,R0               ; R0 IS CRATE
        BIC     #1777,R0            ; CSR ADDRESS
        ADD     #1400,R0
        MOV     #2,(R0)             ; SET F CODE=2
        MOV     (R2),R3             ; GET DATA
        TST     @#HILOL
        BEQ     QTEST               ; 0 IS WHOLE WORD
        BMI     NOSWAP              ; -1 LOWER BYTE
        SWAB    R3                  ; 1 UPPER BYTE
NOSWAP: COM     R3                  ; INVERT DATA (CAMAC BUS IS OPEN COLLECTOR)
        BIC     #177400,R3
        SUB     #200,R3
```

```
QTEST:  MOV     #1,@#QFLAG      ; PRESET FLAG
        BIT     #200,(R0)       ; TEST Q
        BNE     QOK
        CLR     @#QFLAG         ; HERE FOR BAD Q
        MOV     @#ERVALL,R3     ; REPLACE R3 WITH ERVALL
QOK:    MOV     @#STEPL,R0       ; GET STEP TO COUNT
STPLOP: DEC     R0
        BEQ     CLEAN
        TST     (R2)            ; DUMMY READ
        BR      STPLOP
CLEAN:  MOV     @#QFLAG,R0
        JSR     R4,INTSTO
        .WORD   QPTR
        MOV     R3,R0
        JSR     R4,INTSTO
        .WORD   DAPTR
RETURN: RTS     PC
;
FPERMS: TRAP    0
        .ASCIIZ 'FP FMT IN FAST'
        .EVEN
FASTTB:
ADDH:   .WORD   0
ADDL:   .WORD   0
HILOH:  .WORD   0
HILOL:  .WORD   0
STEPH:  .WORD   0
STEPL:  .WORD   0
ERVALH: .WORD   0
ERVALL: .WORD   0
QPTR:   .WORD   0,0,0
DAPTR:  .WORD   0,0,0
QFLAG:  .WORD   0
;
;
;
;                               ;SUBROUTINE TO AID IN INTEGER STORAGE USING
;                               ;'STORE'.IT CLEARS UPPER FAC, AND PUTS R0
;                               ; IN LOWER FAC, THEN STORES FAC IN LOCATION
;                               ;POINTED TO BY THE WORD FOLLOWING THE
;                               ;CALL TO SUBROUTINE
;                               ;LINKAGE THRU R4, WHICH IS PRESERVED
;                               ;R0 IS DESTROYED
INTSTO: MOV     R0,FAC2(R5)
        CLR     FAC1(R5)
        MOV     (R4)+,R0        ;GET STORE ADDRESS INFO TO R0
        JSR     PC,STORE
        RTS     R4
        .END
```

02-JAN-80 COPY OF DX1:OPROOT.MAC, OPTION ILPRINT V1:BB

```
;       DATA AREA FOR FREM, A USER SUBROUTINE FOR BASIC V2,TO ALLOW INSERTION
;       OF MACHINE CODE------- SEE MEMOP2.MAC
;
;       ************************************************************
;       *                                                          *
;       *   MUST BE LINKED IN ROOT SEGMENT IF OVERLAY USED         *
;       *                                                          *
;       ************************************************************
            .TITLE OPROOT
            .GLOBL FIRSTA,LASTA
FIRSTA:       .BLKW 27          ; AREA FREE FOR USER
LASTA:        .WORD 0           ; LAST WORD FREE
;                                ; TOTAL OF 24(10) WORDS
              .END
;
```

```
02-JAN-80 COPY OF DX1:SUBTAB.MAC, OPTION ILPRINT V1:BB

; NEW  FUNCTION TABLE DEFINITION FOR MEMOP2
;*****************************************************
;GOES WITH MEMOP2
        .GLOBL  PEEKST,POKEST,GETBST,STOBST,EXECST,FREEST,FASTST
        .CSECT  BASICR
        .WORD   FUNTAB.
        .CSECT  SUBTAB
FUNTAB:
        .ASCII  'PEEK'
        .WORD   PEEKST
        .ASCII  'POKE'
        .WORD   POKEST
        .ASCII  'GETB'
        .WORD   GETBST
        .ASCII  'STOB'
        .WORD   STOBST
        .ASCII  'EXEC'
        .WORD   EXECST
        .ASCII  'FREM'
        .WORD   FREEST
        .ASCII  'FAST'
        .WORD   FASTST
        .BYTE   0
        .END
```

Table of subroutine names for MACRO additions to BASIC

```
04-JAN-80 COPY OF OLINK.LST, OPTION I     LPRINT V1:BB

DX1:OVRBSC,DX1:OVRMAP=DX1:BASICR,FPMP,DX0:SUBTAB,OPROOT/T/B:400/C
DX1:BASICE/O:1/C
DX1:BASICX,DX0:MEMOPS,DX0:MEMOP2,DX1:GETARG/O:1/C
DX1:BASICH/O:2
```

RT-11 commands used to link new BASIC

APPENDIX C

COMMANDS

There are presently 26 commands recognized by the system. The commands are listed in alphabetical order, each followed by a brief description. A detailed description and example of each command can be found in Sec. 5.7.

A#:	advance through the 2264 buffer memory by executing "#" read commands
AUT#:	start and stop the 2264 digitizer "#" times
CRATE:	set CRATE and STATION numbers
DATA:	set number of points used in graphs (default is 100)
ENQUIRE:	return the present CRATE, STATION, and SUBADDRESS numbers
ERASE:	reset MULTIPLE and OVERLAY flags in graphics routines
F#:	execute CAMAC F codes
GRAPH:	plot data presently stored in 2264 buffer memory
HALT:	stop program, return to BASIC
INITIALIZE:	execute CAMAC Z command
MULTIPLE:	set graphics MULTIPLE flag
NAME:	set operator's name
NEW:	execute stop trigger command sequence for 2264 digitizer
OVERLAY:	set graphics OVERLAY flag
PANEL:	read front panel switch positions on 2264 digitizer
R:	read single data word from 2264 digitizer
READ:	plot data stored on the data disk

SCALE: change scale factors on graphs

SHOT: modify the shot counter

STATUS: return the DCC-11 Control Status Register (CSR) contents

STORE: store data on data disk

SUBADDRESS: set the CAMAC SUBADDRESS

TITLE: add record titles to the TITLE.REC file

V#: read data from channel "#" of 2232A scanning DVM

WIDTH: modify size of next graph

TABLE C-1

Record Titles and Types Contained in TITLE.REC

Type	Title	
010	Function Gen	
020	Cos Pos	
021	Sin Pos	
022	Cos Pos Norm	
023	Sin Pos Norm	
026	V Loop Inner	
027	V Loop Outer	
030	I Plasma	
032	I OH Primary	
033	I Toroidal	
034	I Vertical	
035	I Radial	
040	DI Plasma	(DI indicates
042	DI OH Primary	the time de-
043	DI Toroidal	rivative of I)
044	DI Vertical	
045	DI Radial	
050	NE Fringes	
051	Cos Cpt	
052	Sin Cpt	
053	Folded Fringe	

Table C-1 Continued

Type	Title
103	BZ Wall Cos
114	BZ Scan Sin
113	BZ Scan Cos

Appendix D

TYPICAL DATA ANALYSIS PROGRAM

This program performs analysis on a group of data files, specified in the file WORKLI.ST. These files are selected in advance, with the aid of the program SELECT.BAS, which reads the file CATALO.G on the data disk being used.

The analysis consists of several "passes" over the data. In each pass the time at which a record (record 103) goes through the level "Z2" is determined. For each pass, a new value of Z2 is obtained from "DATA" statements. The values of records 113 and 114 (referred to by N7 and N8) at the time determined above are read, and printed after processing. These values (and others, such as the time of the crossing) are printed on the terminal and also in a disk file "OUTPUT.ANA".

Record 103 is graphed, and for each pass, a vertical line is drawn on the graph, at the time value obtained by the search procedure above. This allows the operator to verify that the search procedure was successful.

LINES	PURPOSE
10 to 2080	Mixture of general purpose code and code specific to this analysis program.
30	enable lower case input
110 to 130	DATA statements specify values for several different "passes" of each file. In this case the numbers are, for line 110:

1st number (100) - voltage threshold (mV)
the program will search for a
crossing of this level
2nd number (680) - begin search for
crossing at 680 μs.
3rd number (1000) - stop search at 1000 μs.

the value 9999 indicates the end of the analysis
of the file, and so the next file is opened, etc.

970 to 990 An example of printing the same information
on to the screen for the operator, and into a
file OUTPUT.ANA for a permanent record;
The function FNF$ spaces the numbers and inserts
commas between them so that other programs can
easily read the numbers from OUTPUT.ANA.
Note that line 940 prints the number of DATA
statements at the beginning of the file
OUTPUT.ANA, so that programs desiring to
read OUTPUT.ANA do not need to know how many
crossing searches were made (= number of DATA
statements).

1000 WORKLI.ST is a file containing the names of the
data files to be analyzed in this run. This
file was created by another program SELECT.BAS,
which looked at the file CATALO.G, on the data
disk, and asked the operator to select the files
to be analyzed.

1030 to 1040	Gets probe position etc. from Parameter file.
1130 to 1140	Finds record types 103, 113, 114 and stores the record numbers in N6, N7, N8 respectively. Type 103 is plotted.
1150 to 1950	Reads data statement, searches for crossing and prints results.
2000 to 2080	Subroutine to find the time(s) when record type 103 passes through the voltage Z2.

SUBROUTINES AND FUNCTIONS

3000 to 3080

Functions:

FNN (T,R) - returns the sample number for record R, given the sample time T

FNB (V,R) - returns the raw data value corresponding to a voltage V in record R

FNT (N,R) - returns the time (in µs) of sample number N in record number R

FNI (N,R) - returns the virtual file index of sample number N in record number R

FNS$ (X,C) - print X, right justified in a field of width C

FNR (X,N) - round X to N decimal places

FNF$ (X,N,C) - print X, rounded to N decimal places, followed immediately by a comma, right justified in a field of width C (including the comma)

FND (N,R) - get sample number N in record R, and convert to millivolts

5000 to 5090	four quadrant Arctangent function; input AsinΘ in S8, AcosΘ in C8 (in radians), result is stored in T8 such that $0 \leq T8 \leq 2\pi$
10010 to 10220	Subroutine to open file, get important information into arrays in memory.
10040 to 10041	it is possible to open the same file as a sequential file and a virtual array file at the same time - saves unnecessary open and close statements
10060	check that the file opened is a data file
10090	check format number; this program only accepts format number 101
10100	position file so that it is ready to read line R3 (see subroutine at 11000)
10110 to 10210	read the directory section (e.g. lines 1001 to 1017 in Fig. 4-7) into arrays
10150	check to see if the line expected (number in L6) was actually read each time
10212 to 10214	read parameter file header information into arrays; Note: Arrays are used to store key parameters so that data can be read more quickly. Otherwise, the program would be continually referring to the disk

10230 to 10290	subroutine to look for record type T4; gives error message if not found; if more than one record has the type T4, the user is informed of this and the first occurrence is chosen; before returning, relevant information about that record is displayed on the terminal, and "J" is set to point to the first data entry in that record
10330 to 10990	subroutine to plot data, using graphics routines; plots only the data appearing in the time range X3 to X5 (in μs) set just before the call in line 1130; X and Y are in physical units (μs, mV); many variations on this section are possible
11000 to 11060	subroutine to allow lines in a sequential file to be accessed randomly if desired; error checking is thorough to relieve the user of this burden; L6 contains the desired line number, and L5 contains the last line number read; one case in which errors cannot be checked (without a considerable loss in speed) is when L6 = L5 + 1; use the subroutine at 11100 to check after reading in this case (preferably in all cases)

11100 to 11110	subroutine to check if reading a line produced the desired line number; L5 should be the first element in the input list of the INPUT statement that read the line, L6 is the expected line number
12000 to 12300	subroutine set designed to access the parameter file information. Entry at 12000 positions the file ready to read line L4 in category T5. Entry at 12200 positions the file at the line in the category C0 belonging to the ID number I0. Entry at 12300 positions the file to read the line associated with data record N4
13000	error recovery (for specific errors only)
17000 to 19910	graphics subroutine
30000 to 30020	prints the output file OUTPUT.ANA on the terminal
31000 to 31050	archives the (temporary) output file

Notes:

- The Parameter file section contains its own format information, and so this program should accept many variations of the Parameter file - eg. explanatory headings omitted, new categories added etc.

In the event that a parameter file structure is generated that does not conform to specifications, data files containing this parameter information can be recognized from their Parameter file ID number (see TYP, Sec. 4.9). Analysis programs detecting these non-standard files can take special action - eg. lines 10216 to 10218 in ANAQ6.BAS for TYP = 1001.

- This program is written in a different BASIC (Multiuser basic V2), so there are slight differences in coding. For example, the virtual arrays in this program are F6$() and F7%(),(not 'VFn').

```
02-JAN-80 COPY OF DX1:ANAQ6.LST, OPTION I LPRINT V1:BB

10 U9=1\ P=4*ATN(1)\ GO=2\ REM: BOTH DIGITIZERS HAVE 2mV RESOLUTION
20 A1=P/2\ BO$=''
30 A=SYS(7,0)
40 REM:
50 ON ERROR GO TO 13000
100 GOSUB 17000\ F9=0\ PRINT C$'WOR 20'\ REM MAKE WS NOW TO AVOID ERASING OUTPUT
102 REM:CROSS,BEGIN,END(INuS)EndwhenDATA=9999
110 DATA 100,680,1000
120 DATA 50,680,1000
121 DATA 0,680,1000
122 DATA -50,680,1000
130 DATA 9999
200 Y3=-200\ Y5=200
900 OPEN 'DX0:OUTPUT.ANA' FOR OUTPUT AS FILE #2, FILESIZE 3
910 L2=0
920 READ Z2\ IF Z2=9999 THEN 930\ L2=L2+1\ READ B3,E3\ GO TO 920
930 RESTORE
940 PRINT #2:FNF$(L2,0,3);FNF$(L2+3,0,3);'Crossing Begin     End (us)  ';
942 PRINT #2:DAT$;'  ';CLK$;'  Radial profile of Bz'
950 FOR I=1 TO L2\ READ Z2,B3,E3\ PRINT #2:FNF$(Z2,2,10);FNF$(B3,2,10);
960 PRINT #2:FNF$(E3,2,10)\ NEXT I\ RESTORE \ PRINT #2:
970 FOR U=0 TO 2 STEP 2
980 PRINT #U:'FILENAME  :RAD. :LEV.:#C:FIRST :LAST T:AVG. T:SIN :COS:CORR.S:';
990 PRINT #U:'PHASE :MAG.    '\ NEXT U
1000 OPEN 'DX0:WORKLI.ST' FOR INPUT AS FILE #6\ DIM #6,F6$(100)=16\ I2=0
1010 RESTORE \ IF F6$(I2)<>'END' THEN 1020\ CLOSE #2\ PRINT 'END***'\ CRASH
1020 IF F6$(I2+1)='N' THEN 1950\ D5$='DX1:'+SEG$(F6$(I2),1,10)\ GOSUB 10010
1030 T4=113\ GOSUB 10230\ IF P0<>1001 THEN 1035
1031 C0=105\ I0=101\ GOSUB 12200\ GO TO 1040\ REM: Fix for PF=1001
1035 GOSUB 12300\ REM: Get the line in the PARM file for TYPE 113
1040 INPUT #1:L5,D1,N3,N3,R1,N3\ PRINT 'Probe ID ';D1\ GOSUB 11100
1130 T4=103\ GOSUB 10230\ N6=N4\ X3=200\ X5=1000\ GOSUB 10330\ REM:PLOT 103
1140 T4=113\ GOSUB 10230\ N7=N4\ T4=114\ GOSUB 10230\ N8=N4
1150 READ Z2\ IF Z2=9999 THEN 1940\ READ N3\ B3=FNN(N3,N6)\ READ N3\ E3=FNN(N3,N6)
1160 J=FNI(0,N6)\ GOSUB 2000\ IF N5<>0 THEN 1170
1162 PRINT D5$;' *NO CROSSINGS THRU';Z2;'FROM ';FNT(B3,N6);' TO ';FNT(E3,N6);
1163 PRINT 'USEC'
1164 S1=0\ C1=0\ S2=0\ A2=0\ M1=0\ Z3=0\ N5=.4\ GO TO 1200\ REM:TAKE CARE OF NO CROSS
1170 Z3=INT(Z1/N5+.5)\ W1=FND(Z3,N6)\ C1=FND(Z3,N7)\ S1=FND(Z3,N8)
1180 S2=-C1*COS(A1)/SIN(A1)+S1/SIN(A1)\ M1=SQR(S2^2+C1^2)
1190 C8=C1\ S8=S2\ GOSUB 5000\ A2=T8
1200 FOR U=0 TO 2 STEP 2
1210 PRINT #U:SEG$(D5$,5,14);',';FNF$(R1,2,6);FNF$(Z2,2,5)FNF$(N5,2,3);
1220 PRINT #U:FNF$(FNT(M4,N6),1,7);FNF$(FNT(M5,N6),1,7);FNF$(FNT(Z1/N5,N6),1,7);
1230 PRINT #U:FNF$(S1,2,5);FNF$(C1,2,4);FNF$(S2,2,7);FNF$(A2,2,7);FNF$(M1,1,7)
1240 NEXT U\ GO TO 1150
1250 REM:
1922 PRINT #U:'USEC'
1940 F8=0\ F9=0\ Q8=Q9+5\ CLOSE #1\ CLOSE 7
1950 I2=I2+2\ GO TO 1010
2000 N5=0\ Z1=0\ FOR I3=B3 TO E3\ IF FND(I3,N6)<>Z2 THEN 2010\ I4=I3\ GO TO 2030
2010 IF FND(I3+1,N6)<>Z2 THEN 2020\ GO TO 2040
2020 IF (FND(I3,N6)-Z2)*(FND(I3+1,N6)-Z2)>=0 THEN 2040\ I4=I3+.5\ GO TO 2030
2030 N5=N5+1\ Z1=Z1+I4\ M5=I4\ IF N5>1 THEN 2040\ M4=I4
2040 NEXT I3\ IF N5<>0 THEN 2050\ PRINT 'No Zeros'\ GO TO 2080
2050 PRINT N5;'Zeros. Min=';F1(N4)+S1(N4)*M4;'uSec.';
2060 PRINT 'Max.=';F1(N4)+S1(N4)*M5;'uSec. Av.=';F1(N4)+S1(N4)*Z1/N5;'uSec.'
2070 X=FNT(Z1/N5,N4)\ Y=Y3\ W9=0\ GOSUB 19005\ Y=Y5\ GOSUB 19005
2080 RETURN
3000 REM: FNS HERE
3010 DEF FNT(N,R)=F1(R)+N*S1(R)\ REM: GIVES TIME FOR SAMPLE N IN RECORD R
3020 DEF FNI(N,R)=N+(B1(R)-1)*256\ REM: GIVES VF INDEX FOR SMP#N IN REC R
3030 DEF FNJ$(X,C)=SEG$(BO$,1,C-LEN(STR$(X)))$STR$(X)\ REM: PAD X TO C PLACES
3040 DEF FNR(X,N)=SGN(X)*INT(ABS(X)*10^N+.5)/10^N\ REM: ROUND X TO N DEC PLACES
3050 DEF FNF$(X,N,C)=FNJ$(FNR(X,N),C-1)$','\ REM:PRT X to N DPs,add',' in C cols
3060 DEF FND(N,R)=GO*G1(R)*F7%(N+(B1(R)-1)*256)\ REM: GET SMPL #N,REC R  MVOLT
3070 DEF FNN(T,R)=INT(.5+(T-F1(R))/S1(R))\ REM: GET SAMPLE # FROM TIME
3080 DEF FNB(V,R)=FNR(V/(GO*G1(R)),0)\ REM: CONV. VOLTAGE BACK TO RAW BIN#
5000 IF C8=0 THEN 5020
5010 T8=ATN(S8/C8)\ GO TO 5050
5020 IF S8<>0 THEN 5040
5030 T8=0\ PRINT 'ATAN2 ERROR'\ RETURN
5040 T8=P/2*SGN(S8)
5050 IF C8<0 THEN 5080
5060 IF T8>=0 THEN 5090
5070 T8=T8+2*P\ RETURN
5080 T8=T8+P
5090 RETURN
10010 REM PROGRAM TO RETREIVE STORED SIGNAL AND PLOT IT
```

```
10020 B2$=CHR$(7)
10040 OPEN D5$ FOR INPUT AS FILE #1
10041 OPEN D5$ FOR INPUT AS FILE #7\ DIM #7,F7Z(10000)
10050 LINPUT #1:L3$
10060 IF SEG$(L3$,1,4)='1001' THEN 10080
10070 PRINT B2$;'BAD FILE. FIRST LINE WAS'\ PRINT L3$\ CRASH
10080 INPUT #1:L5,F6,F1,S5,R2,N3,N3,N3,R3,R4,N3,L1
10090 IF F6=101 THEN 10100\ PRINT 'UNKNOWN PGM WROTE THIS';F6
10100 L6=R3\ GOSUB 11000
10110 FOR I=1 TO R2
10120 INPUT #1:L5,T1(I),C1(I),I1(I),H1(I),L1(I),T1$(I)
10150 GOSUB 11100
10160 L6=L6+1\ NEXT I
10170 L6=R3+R2\ GOSUB 11000
10180 FOR I=1 TO R2\ L6=H1(I)\ GOSUB 11000
10190 INPUT #1:L5,B1(I),F1(I),S1(I),N2(I),E1(I),G1(I),O1(I),O2(I),C2(I),S2(I)
10200 GOSUB 11100
10210 NEXT I\ REM END OF READING DATA HEADER
10211 L6=R4\ GOSUB 11000
10212 INPUT #1:L5,N3,P0,C3,N3,O1\ GOSUB 11100
10213 FOR I=1 TO C3\ L6=O1+R4+I\ GOSUB 11000
10214 INPUT #1:L5,C2(I),O3(I),N3,D1(I)\ GOSUB 11100\ NEXT I
10215 REM: END READING PARM FILE HEADER--NOW CHECK PARM FILE # AND ADJUST
10216 O0=0\ REM:INITIALIZE FIX-UP VARIABLES
10217 IF P0<>1001 THEN 10218\ O0=1\ RETURN
10218 IF P0<>1002 THEN 10219\ RETURN
10219 PRINT 'Unrecognized PARM file #';P0\ CRASH
10230 N3=0\ FOR I=R2 TO 1 STEP -1
10240 IF T1(I)<>T4 THEN 10250\ N3=N3+1\ N4=I
10250 NEXT I\ IF N3=0 THEN 10270\ IF N3=1 THEN 10280
10260 PRINT B2$;N3;'Types';T4;'found'\ GO TO 10280
10270 PRINT B2$;'Type';T4;'not found'\ GO TO 10280
10280 PRINT 'TYP';T1(N4);'F-T';F1(N4);'STP';S1(N4);'#SMP';N2(N4);'L-T';E1(N4)
10290 J=(B1(N4)-1)*256\ RETURN
10330 X7=0\ Y7=0\ REM Linear scales
10340 N3=LEN(D5$)\ X$=SEG$(D5$,N3-5,N3)+' '+T1$(N4)+' (uS)'\ Y$='TYP'+STR$(T4)
10350 GOSUB 18100\ REM GRAPH SPACE
10360 REM:
10410 GOSUB 19200\ REM: SET SCALE FACTORS
10420 W9=0
10430 FOR I=FNN(X3,N4) TO FNN(X5,N4)
10440 REM: PLOT FROM X3 TO X5
10450 X=FNT(I,N4)\ Y=FND(I,N4)
10460 GOSUB 19005\ REM PLOT
10470 NEXT I
10480 GOSUB 18900
10990 RETURN
11000 IF L6<1001 THEN 11002\ IF L6<1001+L1 THEN 11010
11002 PRINT B2$'LINE';L6;'DOESN'T EXIST'\ CRASH
11010 IF L6>L5 THEN 11020\ RESTORE #U9\ L5=1000
11020 IF L6=L5+1 THEN 11050\ FOR D3=L5+1 TO L6-1\ IF END #U9 THEN 11060
11030 INPUT #U9:L5\ IF L5=D3 THEN 11040\ PRINT 'BAD LINE # EXP.';D3;'GOT';L5;
11031 PRINT 'ATTEMPT TO CORRECT'\ GO TO 11010
11040 NEXT D3
11050 RETURN
11060 PRINT 'END OF FILE #';U9\ CRASH
11100 IF L5=L6 THEN 11110\ PRINT 'EXPECTED LINE';L6;',GOT';L5
11110 RETURN
12000 IF C3<>0 THEN 12010\ PRINT B2$'FILE NOT READ YET'\ CRASH
12010 FOR I=1 TO C3\ IF C2(I)=T5 THEN 12030\ NEXT I
12020 PRINT B2$'CAT #';T5;'NOT FOUND IN PARM FILE HEADER'\ CRASH
12030 IF L4<=D1(I) THEN 12060
12040 PRINT B$'ASKED FOR LINE';L4;'IN CAT#';T5;'BUT ONLY';D1(I);'LINES IN FILE'
12050 CRASH
12060 L6=L4+O3(I)+R4+O0\ GOSUB 11000\ REM: THE O0 IS A FIX FOR PF 1001
12070 RETURN
12200 L4=1\ T5=C0\ GOSUB 12000\ FOR I1=1 TO D1(I)\ INPUT #1:L5,N3
12210 GOSUB 11100\ L6=L6+1\ IF N3=I0 THEN 12240\ NEXT I1
12220 PRINT 'ID #';I0;'not found in CAT #';C0;'in the PARM file'\ CRASH
12230 REM:BACK UP ONE LINE
12240 L6=L5\ GOSUB 11000\ RETURN
12300 C0=C1(N4)\ I0=I1(N4)\ GO TO 12200
13000 IF ERL=12200 THEN IF ERR=50 THEN RESUME 12220
13990 ON ERROR GO TO 0\ REM: UNEXPECTED ERROR
17000 REM START UP ROUTINE
17010 REM F7 IS ERASE GRAPHICS FLAG,F8 IS ERASE WORKSPACE,F9 IS ERASE ALL
17020 F7=1\ F8=1\ F9=1
17030 REM SET COMMAND CHARACTER IN C$ SO THAT IT DOESNPEAR IN TEXT
17040 C$=CHR$(33)
17050 REM SET STANDARD LINE TYPE
17060 PRINT C$;'LIN 1'
```

```
17070 REM TOP L.H. CORNER OF GRAPHICS AREA DEFAULTS TO ROW 3,COL 7
17080 Q8=3\ P8=7
17090 REM DEFAULT LABELS
17100 X$='Put X label in X$'
17110 Y$='Put Y label in Y$'
17120 REM SETS UP DEFAULT SIZE GRAPH BOX AND SCALE FACTOR
17130 REM X4=LEFT,X6=RIGHT X,Y4=BOTTOM Y,Y6=TOP Y(ALLSCREENUNITS)
17140 X4=0
17150 X6=119
17160 Y6=111
17170 Y4=0
17180 REM
17190 RETURN
18100 IF F7=0 THEN 18290\ REM SET UP GRAPHICS AREA-ERASE IF F8,REDEFINE IF F9
18110 Q9=Q8+INT(Y6/14)
18120 P9=P8+INT(X6/8)
18130 IF Q8<0 THEN 18150
18140 IF Q8>Q9 THEN 18150
18145 GO TO 18170
18150 PRINT 'ERROR IN VERTICAL SIZE OR POSITION Q8=';Q8;'Q9=';Q9
18160 STOP
18170 IF P9>80 THEN 18190
18175 IF P8<1 THEN 18190
18180 IF P9<P8 THEN 18190
18185 GO TO 18230
18190 PRINT 'ERROR IN HORIZONTAL SIZE OR POSITION P8=';P8;'P9=';P9
18200 STOP
18210 REM NOW TEST ERASE FLAGS
18230 IF F9=0 THEN 18250
18233 IF Q9+3<31 THEN 18240
18234 PRINT C$;'WOR 20'\ REM DEFINE A SMALLER ONE
18235 PRINT C$;'MON H;ERROR--ATTEMPT TO DEFINE WS>30';CHR$(7)
18237 GO TO 18250
18240 PRINT C$;'WOR';Q9+3
18250 IF F9=0 THEN 18270
18260 PRINT C$;'ERA W'
18270 PRINT C$;'GRA ';Q8;Q9;P8;P9
18280 P7=P8-1
18282 IF (P9-P8+1)*(Q9-Q8+1)<2200 THEN 18290
18285 PRINT CHR$(7);'WARNING--GRAPH WILL USE MOST OF THE 8K DISPLAY RAM'
18290 RETURN
18300 REM LABELS,NUMBERS
18305 PRINT C$;'WOR H'
18310 PRINT C$;'JUMP ';Q9+3;INT((P8+P9-LEN(X$))/2)
18320 PRINT X$;
18330 IF P7>1 THEN 18340
18333 P7=2
18340 FOR I9=1 TO LEN(Y$)
18345 PRINT C$;'JUM ';INT((Q8+Q9-LEN(Y$))/2)+I9;P7-1
18350 PRINT SEG$(Y$,I9,I9)
18370 NEXT I9
18373 P7=P7-2
18375 PRINT C$;'MON H'
18380 RETURN
18400 REM AUTO TICKS AND SCALES---DO X AXIS FIRST
18405 PRINT C$;'WOR H'
18410 P1=P4\ Q1=Q4\ REM TICK START AT AXIS CROSSING
18415 Q3=0\ REM TURN OFF Y TICKS
18417 IF X7=1 THEN 18480
18420 P2=10^INT(LOG(ABS(X5-X3))/LOG(10))\ REM LARGEST POWER OF TEN THAT FITS
18430 P2=P2/10\ P3=1\ REM MINOR DIVISIONS
18440 GOSUB 19400
18443 P2=10*P2
18445 IF P2<.5*ABS(X5-X3) THEN 18460
18450 P2=P2/2
18460 P3=-2
18465 GOSUB 19400
18470 GO TO 18640\ REM NOW FOR Y AXIS
18480 P2=10\ P3=-2\ REM MAJOR DIVISIONS----LABEL THEM
18490 GOSUB 19400
18500 P3=1
18600 FOR I7=2 TO 9
18610 P1=I7*P4
18620 GOSUB 19400
18630 NEXT I7
18640 REM NOW Y TICKS
18650 P3=0\ REM TURN OFF X AXIS
18660 IF Y7=1 THEN 18750
18670 Q2=10^INT(LOG(ABS(Y5-Y3))/LOG(10))\ REM LARGEST POWER OF TEN THAT FITS
18680 Q2=Q2/10\ Q3=1\ REM  MINOR DIVISIONS
18690 GOSUB 19400
```

```
18700 Q2=10*Q2
18710 IF Q2<.5*ABS(Y5-Y3) THEN 18730
18720 Q2=Q2/2
18730 Q3=-2\ REM MAJOR DIVISIONS---NUMBER THEM
18735 GOSUB 19400
18740 GO TO 18820
18750 Q2=10\ Q3=-2\ REM MAJOR DIVISIONS----NUMBER THEM
18760 GOSUB 19400
18770 Q3=1
18780 FOR I7=2 TO 9
18790 Q1=I7*Q4
18800 GOSUB 19400
18810 NEXT I7
18820 RETURN
18900 GOSUB 19300\ REM AXES
18910 GOSUB 18400\ REM TICKS,NUMBERS
18920 GOSUB 18300\ REM LABELS
18930 RETURN
19000 REM CONVERT FROM DATA TO SCREEN COORDINATES
19002 W9=2
19005 REM ENTER HERE IF POINT IS TO BE OUTPUT AS WELL
19006 REM IF W9=0 THEN MOVE TO XY, THEN SET W9=1, DRAW FROM OLD XY TO NEW XY
19008 IF X7=1 THEN 19030
19010 X1=INT((X-X3)*X8+X4+.5)
19020 GO TO 19040
19030 X1=INT(LOG(X/X3)*X8+X4+.5)
19040 IF Y7=1 THEN 19070
19050 Y1=INT((Y-Y3)*Y8+Y4+.5)
19060 GO TO 19080
19070 Y1=INT(LOG(Y/Y3)*Y8+.5)
19080 IF W9=1 THEN 19130
19085 IF W9=2 THEN 19120\ REM W9=2 IS A CONVERSION ONLY
19090 W9=1
19100 X2=X1
19110 Y2=Y1
19120 RETURN
19130 PRINT C$;'VEC';X2;Y2;X1;Y1
19140 GO TO 19100
19200 REM SET UP NEW SCALE FACTORS X8, Y8
19204 P4=X3\ Q4=Y3\ REM SET ORIGIN AT FAR LEFT BOTTOM CORNER
19210 IF X7=1 THEN 19240
19220 X8=(X6-X4)/(X5-X3)
19230 GO TO 19250
19240 IF X5*X3<>0 THEN 19248
19243 PRINT C$;'MON H;LOG ERROR LEFT X=';X3;'RIGHT X=';X5
19245 STOP
19248 X8=(X6-X4)/LOG(X5/X3)
19250 IF Y7=1 THEN 19280
19260 Y8=(Y6-Y4)/(Y5-Y3)
19270 GO TO 19290
19280 IF Y3*Y5<>0 THEN 19288
19283 PRINT C$;'MON H;LOG ERROR BOTTOM Y=';Y3;'TOP Y=';Y5
19285 STOP
19288 Y8=(Y6-Y4)/LOG(Y5/Y3)
19290 RETURN
19300 REM       DRAW AXIS
19320 X=P4
19330 Y=Q4
19340 GOSUB 19000
19350 REM W9=2 WILL CAUSE NO PLOTTING ACTION, JUST CONVERSION
19360 PRINT C$;'VEC';X4;Y1;X6;Y1
19370 PRINT C$;'VEC';X1;Y4;X1;Y6
19380 RETURN
19400 REM  TICK DRAWING-ENTER HERE FOR AUTOMATIC TICK SIZE
19405 PRINT C$;'WOR H'
19410 REM  CALCULATE P5, Q5, THE TICK SIZE IN SCREEN UNITS
19420 P5=INT(((ABS(Y6-Y4)/100)+2)*P3)
19430 Q5=INT(((ABS(X6-X4)/100)+2)*Q3)
19440 REM    ENTER HERE TO USE YOUR OWN TICK SIZE
19450 REM    ASSUME STARTING POINT  INSIDE WINDOW
19455 IF P5=0 THEN 19660
19460 REM    FIND TICK NEAR THE EDGE
19470 Y=Q4
19475 FOR I9=0 TO 800
19480 IF X7=1 THEN 19500
19485 X=P1+I9*P2
19490 GO TO 19510
19500 X=P1*(P2^I9)
19510 GOSUB 19000
19520 IF X1>X6 THEN 19550
19530 IF X1<X4 THEN 19550
```

```
19540 NEXT I9
19550 REM     NOW DO TICKS BACKWORDS
19560 FOR I8=I9-1 TO -800 STEP -1
19565 IF X7=1 THEN 19580
19570 X=P1+I8*P2
19575 GO TO 19590
19580 X=P1*(P2^I8)
19590 GOSUB 19000
19600 IF X1>X6 THEN 19660
19610 IF X1<X4 THEN 19660
19620 PRINT C$;'VEC';X1;Y1+P5;X1;Y1;X1;Y1-P5
19633 IF P5>0 THEN 19640
19635 PRINT C$;'JUMP 'Q9+1;P8+INT((X1-12)/8)
19637 PRINT X;
19640 REM TWO PRINTS CORRESPOND TO ABOVE & BELOW AXIS, THIS WAY SOMTHING GETS
19645 REM DRAWN --EVEN AT THE EDGE OF THE GRAPHICS AREA
19650 NEXT I8
19660 REM     NOW FOR Y AXIS
19665 IF Q5=0 THEN 19910
19670 X=P4
19680 FOR I9=0 TO 800
19690 IF Y7=1 THEN 19720
19700 Y=Q1+I9*Q2
19710 GO TO 19740
19720 Y=Q1*(Q2^I9)
19740 GOSUB 19000
19750 IF Y1>Y6 THEN 19780
19760 IF Y1<Y4 THEN 19780
19770 NEXT I9
19780 REM     AFTER HAVING FOUND EXTREME TICK POSITION, DRAW TICKS
19790 FOR I8=I9-1 TO -800 STEP -1
19800 IF Y7=1 THEN 19830
19810 Y=Q1+I8*Q2
19820 GO TO 19850
19830 Y=Q1*(Q2^I8)
19850 GOSUB 19000
19860 IF Y1>Y6 THEN 19910
19870 IF Y1<Y4 THEN 19910
19880 PRINT C$;'VEC';X1+Q5;Y1;X1;Y1;X1-Q5;Y1
19893 IF Q5>0 THEN 19900
19895 PRINT C$;'JUMP ';Q9-INT(Y1/14);P8-LEN(STR$(Y))
19897 PRINT STR$(Y)
19898 IF P8-LEN(STR$(Y))-1>P7 THEN 19900
19899 P7=P8-LEN(STR$(Y))-1
19900 NEXT I8
19905 PRINT C$;'MON H'
19910 RETURN
30000 CLOSE \ OPEN 'OUTPUT.ANA' AS FILE #1\ PRINT \ PRINT
30010 IF END #1 THEN 30020\ LINPUT #1:L$\ PRINT L$\ GO TO 30010
30020 CLOSE \ STOP
31000 PRINT 'Insert a non-data disk into DX1: if you don't want to save the '
31010 PRINT 'data on the system disk.'
31020 PRINT 'new file name (include DX0: or DX1)  ';\ INPUT D1$
31030 CLOSE \ OPEN D1$ FOR OUTPUT AS FILE #1, FILESIZE 30
31031 OPEN 'OUTPUT.ANA' FOR INPUT AS FILE #2
31040 IF END #2 THEN 31050\ LINPUT #2:L$\ PRINT #1:L$\ GO TO 31040
31050 CLOSE \ PRINT 'Replace data disk'\ STOP
```